SIGRAD 2007

The Annual SIGRAD Conference

Special Theme: Computer Graphics in Healthcare

November 29–30, 2007

Uppsala, Sweden

Conference Proceedings

Organized by

SIGRAD, svenska lokalavdelningen av Eurographics

and

Uppsala University

Edited by

Anders Hast

Published for

SIGRAD, svenska lokalavdelningen av Eurographics

by Linköping University Electronic Press

Linköping, Sweden, 2007

Linköping Electronic Conference Proceedings, No. 28
Linköping University Electronic Press
Linköping, Sweden, 2007

ISBN 978-91-7393-990-4
ISSN 1650-3686 (print)
http://www.ep.liu.se/ecp/028/
ISSN 1650-3740 (online)
Print: LiU-Tryck, Linköpings universitet, 2007

Cover illustrations taken from various papers in these proceedings.
Painting of a university building by Jakob Nisell.

Table of Contents

Preface

These proceedings contain the papers from the SIGRAD 2007 conference which was held on the 29th and 30th of November in Uppsala, Sweden. The topic of this year's conference is Computer Graphics in Healthcare. As in previous years, we also welcome paper submissions in various other graphics areas.

The SIGRAD conference has an explicit ambition to broaden its geographic scope beyond the national borders of Sweden. We are therefore very happy to have several international contributions this year. The keynote speakers this year are Dirk Bartz from the University of Leipzig and Stefan Seipel from Uppsala University and we would like to thank them both for their interesting talks. The topic of the Dirk's keynote is "Visual Computing for Medicine" and the topic for Stefan's keynote is "How to (and not to) fool your brain to perceive 3D".

We would also like to thank the program committee that provided timely reviews, and helped in selecting the papers for these proceedings.

Many thanks to our generous sponsors: The Virtual IT faculty at Uppsala University and UPPMAX.
We wish all participants a stimulating conference, and hope they take the chance and to create new connections in the European graphics community.

Anders Hast
Program Chair SIGRAD 2007

SIGRAD 2007

The SIGRAD 2007 IPC committee consisted of experts in the field of computer graphics and visualization from several European countries. We thank them for their comments and reviews.

Program Chair
Anders Hast, University of Gävle

Conference Organizing Committee Members
Anders Hast, University of Gävle
Kai-Mikael Jää-Aro, Telestream AB
Stefan Seipel , University of Gävle

International Program Chairs
Ewert Bengtsson, Uppsala Univeristy
Matt Cooper, Limköping University
Modris Dobelis, Riga Technical University
Anders Hast, University of Gävle
Kai-Mikael Jää-Aro, Telestream AB
Ivana Kolingerova, University of West Bohemia in Pilsen
Lars Kjelldahl, Royal Institute of Technology
Thomas Larsson, Mälardalen University
Ingela Nyström, Uppsala University
Lennart Ohlsson, Lund University
Stefan Seipel , University of Gävle
Krzysztof Tytkowski, Silesian University of Technology
Anders Ynnerman, Linköping University

The SIGRAD Board for 2007
Kai-Mikael Jää-Aro, Chair
Thomas Larsson, Vice Chair
Lars Kjelldahl, Treasurer
Anders Hast, Secretary
Stefan Seipel, Member
Örjan Wretblad, Member
Anders Ynnerman, Member
Anders Backman, Substitute
Alex Olwal, Substitute

Visual Computing for Medicine

Dirk Bartz

University of Leipzig, Germany

Abstract

Medical visualization has been a long way down the road from the first - by today's standard - crude images to the current sophisticated rendering results. However, for a successful application of visualization, we need to look at the whole visual computing pipeline, which includes image processing, visualization, interaction, and in addition into perceptual issues of visual computing.
In my talk, I will discuss certain aspects of this pipeline. Starting from early image filtering after image acquisition (which by itself is also part of the medical imaging/image processing stage), segmentation is need to identify specific organs, in particular if they cannot be identified by standard classification approaches. After the preparation of the potentially multi-value and multi-field (modal) structured datasets, they are visualized using the whole variety of direct and indirect volume rendering approaches. Here, I will demonstrate that both approaches have advantages and disadvantages, and hence their place in medical visualization. Finally, we need to interact with the resulting renderings either pre-(post-)operatively for planning purposes, or in an intra-operative environment during an intervention. At all these stages, the intermediate and final results are interpreted mostly through the visual system of humans. Hence, we also need to consider how perception is not only influencing the interpretation, but also how we can improve it.

How to (and not to) fool your brain to perceive 3D

Stefan Seipel

Uppsala University, Sweden

Abstract

The use of “advanced” 3D visualizations has become standard in many scientific, commercial and entertainment applications. Much of the term “advanced” is related to the techniques that have emerged during the years to exploit human’s capability of perceiving depth from binocular vision. While technical solutions are readily available to recreate high-quality stereo-graphic images on planar displays, there are perceptual and physiological limits to our sensation of true spatial images. In addition, there are other efficient mechanisms that help the observer to extract the 3D layout of a perceived scene. In this tutorial I will give an introduction into stereo-graphic display techniques and I will discuss perceptual limits, common pitfalls and trade-offs in generating stereographic images. In the presentation I will give hints and tips on how to generate and display 3D images.

Using a GPU-based Framework for Interactive Tone Mapping of Medical Volume Data

Matthias Raspe* Stefan Müller†

Computer Graphics Working Group
Institute for Computational Visualistics
University of Koblenz-Landau

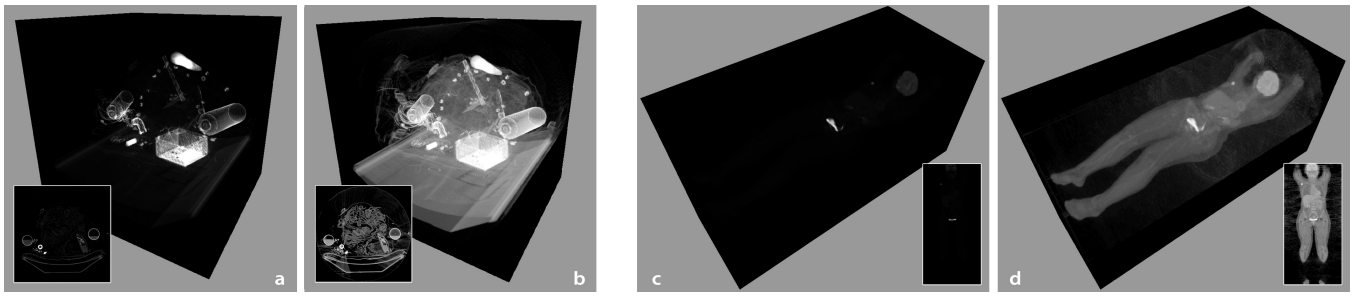


Figure 1: *Direct volume rendering of high dynamic range volume data. The images on the left show a CT data set, on the right a PET scan of a human body, with insets depicting a slice of the according volumes for reference. While the linear mapping (a, c) reveals only the highest values of the data sets, tone mapping algorithms (b,d) can display the whole dynamic range in real-time.*

Abstract

Medical workstations nowadays visualize large amounts of data from image acquisition systems that have dynamic ranges usually much higher than standard devices can display. In order to examine the data or control other processing steps, the user specifies windowing parameters to map the input values to the displayable output range. While this operation can be performed efficiently even on large datasets by using simple lookup tables, no acceptable performance is achieved when advanced algorithms from high dynamic range imaging are needed. Especially data from functional imaging modalities has a much higher dynamic range and requires considerable interaction for proper visualization using the traditional windowing approach. Therefore, we propose to integrate tone mapping algorithms into the visualization pipeline of volume data by exploiting modern graphics hardware. To allow for a flexible implementation and integration with other processing steps, we will present our programming framework and compare the performance to CPU implementations. In addition, we will discuss different tone mapping approaches in consideration of miscellaneous medical modalities and the role of transfer functions in the context of high dynamic range rendering.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms; I.4.3 [Image Processing and Computer Vision]: Enhancement—Grayscale manipulation; I.4.10 [Image Processing and Computer Vision]: Image Representation—Volumetric

Keywords: graphics hardware, volume data processing, medical visualization, tone mapping

1 Introduction

During the last years, programmable graphics hardware (GPU) has become more important in different fields of applications [Owens et al. 2007]. This is mainly due to the high computing power clearly exceeding that of modern processors (CPUs) and thus offering a much better price-to-performance ratio (see figure 2). Driven by the growing entertainment market, the performance of the GPU's inherently parallel and integrated architecture can be increased at a fast pace, making them especially attractive for data intensive tasks such as image processing, visualization, simulation, etc. Utilizing the graphics hardware not only for visualization is of particular interest, because the data being processed can be displayed without additional conversion or transfer, contrary to visualizing results of CPU algorithms.

However, there are some drawbacks with this development. First, using the graphics hardware for general purposes still involves graphics programming and thus thorough knowledge in computer graphics. In addition, complex algorithms are not easily ported to the GPU due to limitations in memory access, programming constructs, etc. This fact is even aggravated by the limited graphics memory which is usually much smaller than the host system's memory. As a result, much communication between the GPU and the host is needed, being a typical bottleneck especially for memory intensive tasks. In order to investigate the benefit of commodity hardware for "real world" applications, we have built the cross-platform programming environment "Cascada". The main intention of this system is realizing volume processing and visualization algorithms on GPUs, focused on (but not limited to) medical vol-

*e-mail:mraspe@uni-koblenz.de
†e-mail:stefanm@uni-koblenz.de

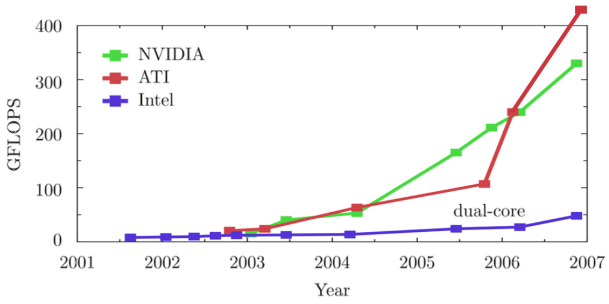


Figure 2: *Development of raw performance for commodity graphics hardware compared to CPUs. ([Owens et al. 2007])*

ume data. As GPUs are optimized for two-dimensional textures consisting of four channels, however, we use a compact volume representation in order to exploit the graphics hardware's capabilities. This also allows for loading and working with volumes larger than the maximum texture size and is in addition compatible with older graphics systems. These textures are updated using an on-demand scheme, i.e., data is converted and transferred to the currently needed representation only once. Thus, interchanging GPU and CPU algorithms as part of a whole sequence processing the same data is simple, yet achieving the best performance for data transfer between the two devices. These sequences resemble one layer of our hierarchical representation of workflows.

As the data acquired and processed usually has a higher value range than standard output devices, different approaches can be applied to map the range of values to the display. For medical data, controlling a window of varying width and center that defines a linear mapping within this window is the de-facto standard in examining image data. While this is straightforward to use and allows for fast implementations, its results are limited and often require lots of manual interaction. Depending on the focus of the diagnosis this can become a tedious manual process.
In the field of computer graphics and computer vision, high dynamic range images are a common approach to represent simulated or captured illumination without introducing errors due to band-limiting the signal. These images are stored in data types of a larger numerical range and/or precision, typically 16 or 32 bit integers or floating point. However, displaying such HDR data by mapping the input data linearly to the output device's range is not sufficient, as only the bright areas would be visible (see figure 3). To overcome this issue, a lot of research has been done in both the computer graphics and computer vision community. With such a foundation it is quite obvious to apply those techniques to medical data in order to improve the visualization. As each modality in medical image acquisition requires a different interpretation – depending on the protocol even single acquisitions – this topic has to be discussed more thoroughly to avoid misinterpreting the results.

The remainder of the paper is structured as follows: In the next section we will discuss existing approaches, focusing on both GPU implementations and tone mapping algorithms. In section 3 we will present our system by outlining the environment with details about our abstract representation of procedures, data handling, the modular concept for shader implementation, etc. Also, we will discuss different tone mapping algorithms and already look at their possible application to different modalities. The results of applying the techniques to medical volume data will be described in section 4 with a discussion of both performance and visual results. We will conclude and propose directions of further investigation in the last section.

2 Related work

In this section we will discuss existing approaches and techniques to outline the context of this paper. Therefore, graphics hardware developments are summarized, followed by a more closer look on tone mapping methods.

2.1 General Purpose GPU

Using commodity graphics hardware for non-graphic application has become an important research topic for several years. This is mainly due to the rapid advances in hardware and programming capabilities, allowing much more flexible programs (so-called "shaders") with unprecedented computing performance. In their survey, Owens et al. give an exhaustive overview of today's technology and its use in all kinds of applications [Owens et al. 2007]. Quite recently, graphics hardware companies have developed dedicated hardware and APIs for non-graphical applications [NVI 2007], [Hensley 2007] to overcome the graphics-only programming concepts.
Although this trend is promising and will continue at an increase rate clearly outperforming the development of current (multi-core) CPUs, as depicted in figure 2, applications often suffer from limited bandwidth for the data transfer to and especially from the graphics memory. However, Langs et al. [Langs and Biedermann 2007] have shown recently that for example advanced filtering of large video data can still be performed several orders of magnitudes faster than CPU-only implementations.

2.2 GPU programming

Aside from the hardware architecture itself, programming such devices has also become more important. High-level languages like Cg [Fernando and Kilgard 2003] or GLSL [Rost 2005] have become the de-facto standard for GPU programming. Although this already enables more flexible and rapid development of shaders, approaches aiming at another abstraction layer have been proposed. McCool and others [McCool et al. 2002], [McCool et al. 2004] have developed "meta-languages" to integrate shader functionality directly into the application code, with concepts like shader algebra and a separation of frontend and backends for different platforms. In addition, Buck et al. [Buck et al. 2004] have developed a streaming-oriented extension that regards operations as kernels applied to data, with according representations for the data, parameters, etc. on the graphics hardware.
Yet another approach is the representation of shader functionality as directed graph, going back to concepts like Cook's shade trees [Cook 1984] that have been used in almost all systems for computer generated images – even in the pre-GPU era. Using this method, complex shaders can be constructed from rather simple and optionally shared components. Related approaches from Abram et al. [Abram and Whitted 1990] or, more recently, McGuire et al. [McGuire et al. 2006] regard shaders as building blocks, emphasizing the specific features of shader programs like different types or input parameters.

2.3 High dynamic range imaging

Mainly initiated by the seminal work of Debevec et al. [Debevec and Malik 1997], a lot of research on high dynamic range imaging, i.e., image data with luminance values of multiple orders of magnitude has been done since then. However, such HDR data is usually displayed on devices with a much lower dynamic range. Although first prototypes of HDR displays are available, so-called tone mapping (or tone reproduction) algorithms still need to be applied to visualize the data adequately. For reference, Reinhard et al. review

existing approaches and address the whole ”pipeline” of high dynamic range imaging in their book [Reinhard et al. 2005].
Basically, tone mapping operators can be categorized into *global* or *local*, some of them with an additional time-dependency. Global operators define some function that maps equal input values to equal output values, thus being computationally inexpensive. These functions can be as simple as a linear function, whereas more advanced operators are more complex and usually incorporate a logarithmic term and other properties of human perception. The results of such tone mapping functions are illustrated in figure 3 and show the already high potential of global operators.

Figure 3: *Different global operators applied to an HDR scene: linear (a), logarithmic (b), and exponential scaling (c), Reinhard's operator (d) (Images courtesy [Reinhard and Devlin 2005])*

In contrast to the ”constant” mapping, local operators are able to adapt to changes by considering the neighborhood of each value. While this approach is very powerful, applying such locally varying operators to medical data needs some discussion. Lately, Bartz et al. [Bartz et al. 2006] have proposed to use tone mapping operators on medical data for improved visual representation. Their algorithm is based on the local operator from Reinhard et al. [Reinhard et al. 2002] with an additional variant for regarding a three-dimensional neighborhood, as is advantageous for most volumetric data. Due to the local nature of the algorithm, the processing time of several seconds for moderately sized volumes is obviously not interactive. They also use only datasets of traditional modalities as CT and MRI, the former providing even a constant mapping (i.e., Hounsfield units) of the measured values. More involved data like PET or non-scalar MRI from functional or diffusion-tensor imaging, that cannot be tone mapped locally without introducing errors has not been considered or left as future work.
Finally, several work has been done to implement the techniques on graphics hardware, as image processing is a particularly suitable application for GPUs. In 2003 already, Goodnight et al. [Goodnight et al. 2003] have successfully realized a time-dependent (i.e., adapting over time, thus mimicking the human visual system) tone mapping system for color images. Due to the much lower performance and more restricted programming of graphics hardware back then, they have not been able to achieve real-time frame rates. Vollrath et al. [Vollrath et al. 2005] have proposed a generic, real-time capable framework to implement the volume rendering pipeline on the graphics hardware. Their system implements Reinhard's tone mapping operator successfully, but they do not discuss it in the context of medical data. Another representative work is that of Yuan et al. [Yuan et al. 2006] who have developed a sophisticated system for visualizing large high dynamic range data volumes at interactive rates. The 3D and 4D data mainly from numerical simulations, geosciences, etc. is processed with especially handling precision issues. However, they also have not investigated in the applicability of their methods on medical data sets.

3 Technical overview

In this section we will explicate the technical details of our contribution. After describing our programming framework and discussing some of its key features related to GPU-accelerated volume processing and visualization, we will outline the tone mapping algorithms we have used for our experiments. In addition, we will discuss the different modalities of medical data with respect to tone mapping operators.

3.1 The GPU-based system ”Cascada”

Starting with a project on segmenting medical data, we have developed a system for (GP)GPU programming called ”Cascada”. This cross-platform framework focuses on processing (medical) volume data by applying modular algorithms implemented as shader programs, i.e., on graphics hardware. Using the GPU for such computation is highly motivated by the rapid performance increases in contrast to CPUs, as shown in the introduction of this paper. However, the raw performance of the graphics hardware does not include all other steps like data transfer, shader handling etc. To this end, our framework is also used as platform for comparing different types of algorithms aiming at general categories where the GPU is preferable or where the overhead might outweigh the performance gain.

3.1.1 Hierarchical representation of functionality

In order to abstract from the graphics programming details, algorithms are represented hierarchically (see figure 4): so-called sequences encapsulate procedures that can range from simple thresholding to more complex operations like region growing. Sequences in turn consist of multiple passes, i.e., drawing geometry with assigned shader programs, usually to offscreen buffers to allow for advanced processing. These passes can resemble single operations or multiply run passes, controlled by a fixed number of iterations or until some condition is met (e.g., region growing has converged). The output of one pass is then used as input to the subsequent pass by setting the texture parameters accordingly. Finally, shader programs resemble objects containing a GLSL vertex and fragment program, together with an automatic infrastructure for handling uniform parameters on both the CPU and GPU efficiently, concatenation of shaders, etc. [1]

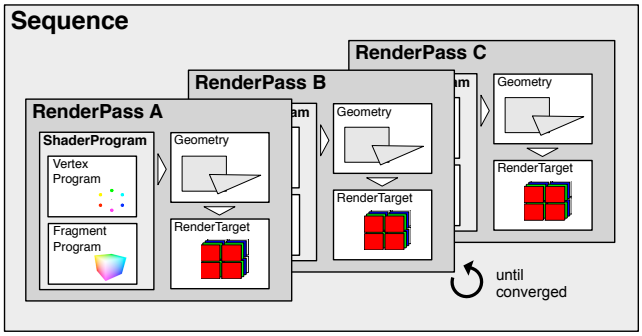


Figure 4: *Hierarchy of rendering components in Cascada. Sequences of render passes consist of shader programs which are applied to geometry. The passes can be controlled by conditions and render into targets used as inputs for subsequent passes, read from for CPU operations, etc.*

[1] It should also be noted that by utilizing the composite pattern, sequences can contain sequences themselves, thus allowing for even more flexible designs.

3.1.2 Internal data handling

The data itself is represented as volumes packed into RGBA-tuples, thus allowing for direct rendering into the volume and exploiting the SIMD architecture of GPUs. Therefore, four successive slices of the scalar volume (or four DICOM slices) are combined into one RGBA slice. For both backward compatibility and better performance, the system uses aside from native 3D textures a two-dimensional representation of the volumetric data as introduced by Harris et al. [Harris et al. 2003]. As shown in [Langs and Biedermann 2007], the time for accessing the flat-3D texture by converting the 3D texture coordinates into the 2D address is even less than direct 3D access. This scheme is depicted in figure 5.

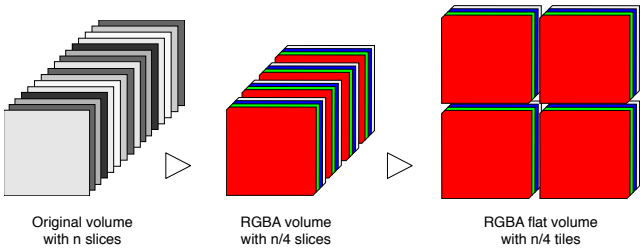


Figure 5: *Representation of volumes in Cascada: The scalar volume is "compressed" into RGBA channels and finally spread into a 2D RGBA texture to be also used as render target.*

Cascada also provides CPU equivalents of the aforementioned sequences and therefore allows for transferring data between graphics and main memory. To this end, Cascada uses a "lazy evaluation" policy to avoid unnecessary bus communication for the complete volume. As the CPU versions of the algorithms are not optimized to leverage parallelism with SIMD units and instructions or utilizing multiple cores, they serve as reference with respect to the results rather than performance.

3.1.3 Visualization and interaction

At the level of render passes in our hierarchical representation, the system does not distinguish between processing and visualization steps. The only difference is the target of the pass/sequence (i.e., offscreen/onscreen buffer) which can be changed during runtime, if needed. Therefore, "Cascada" implicitly allows for displaying the results of algorithms while they are computed, with a negligible performance overhead on modern graphics hardware. Currently, our system provides several volume visualization modes:

- Standard multi-planar reformation display with slices movable in the main directions
- GPU ray casting with different compositing modes (X-ray, maximum intensity projection, isosurface rendering)
- Direct volume rendering with basic transfer function support
- Integration of geometric primitives (glyphs, lines, color coding) in volume rendering (work in progress)

Together with the support for additional interaction devices (3D mouse, haptics devices) it is thereby possible to literally interact with the algorithms: "drawing" parameters for ray casting or controlling iterative algorithms are two examples for the advantage of computing directly on the graphics hardware.

3.1.4 Performance examples

In order to provide some measure for the efficiency of our framework, we have compared its processing performance with MeVisLab [MeVis 2007], a widely used software system for efficient (medical) data processing and visualization. Most of MeVisLab's core is based on a sophisticated image processing library with high performance even for very large data sets. Additionally, the modular system of the components in combination with the extremely powerful, graph-oriented frontend enable the user to develop applications rapidly. However, the system does not utilize the graphics hardware except for visualization purposes and graphical shader programs.

To give some idea of the performance relations between Cascada and CPU-only implementations, we have set up some scenarios of increasing computational complexity. We will provide both the times for the pure GPU pass, and the time needed for data transfer to and from the video memory for better comparison. Our system has been an Intel Core2Duo (2.4 GHz) with 2 GB RAM and an Nvidia Geforce 8800 GTS with 640 MB RAM, running on Windows XP. For MeVisLab we have used the latest SDK version (1.5.1) available from the website with default settings. We have applied the algorithms in both applications on the same datasets: a $512 \times 512 \times 223$ CT scan loaded from DICOM files (first value), and an MRI scan $256 \times 256 \times 256$ as raw file (second value).

Operation	MeVisLab	Cascada[1]	Cascada[2]
Binary	0.73 / 0.21	1.9 / 0.55	0.038 / 0.011
Gradient 2D	10.1 / 2.9	2.6 / 0.7	0.06 / 0.017
Gradient 3D	14.9 / 3.9	2.6 / 0.72	0.059 / 0.018
Gauss 2D	1.36 / 0.37	2.48 / 0.73	0.061 / 0.017
Gauss 3D	3.65 / 1.01	2.59 / 0.78	0.09 / 0.026

Table 1: *Average computation times (in seconds) for applying operations of different complexity on the two data sets.* [1] *Timing including data transfer to and from the GPU;* [2] *Time for image operation only, but including shader and handling.*

The results show a clear advantage of the GPU version over the CPU implementation, with a speedup factor of up to 250. However, the impact of the data transfer on the performance outweighs the gain for simple operations (binary threshold) or filters with small neighborhood (Gauss 2D). These timing give only some coarse impression of the results, with Cascada being not fully optimized with respect to data transfer yet.
In another test run we have compared a sequence consisting of algorithms of different complexity (incl. iterative operations) to evaluate the relevance of data transfer, leaving the GPU (including all overhead) at least one order of magnitude faster than the software implementation. Therefore, our proposition of utilizing the graphics hardware for image processing operations is highly advantageous has been confirmed, even with the additional rendering infrastructure. However, not all operations have been considered yet, especially where the GPU suffers from its architectural limitations. We are still investigating in this to allow for a categorization of different algorithm types and be able to decide whether a GPU or CPU implementation is more suitable over the other – even at run time with our system already supporting the mixed use of both platforms.

3.2 Medical data

For medical diagnostics and, of course, depending on the current problem, there are lots of different imaging modalities available, with the majority being tomographic data nowadays. Dedicated hardware and software integrated in those systems allow for rapid reconstruction of the measured data resulting in slices of usually scalar values. These slices have certain geometric properties (e.g.,

thickness, distance, resolution) and can thus be regarded as a volume representing the examined anatomy. That is, we assume a regular grid of voxels with each voxel containing a value at a certain point in space. These values are usually scalar values, i.e., gray values within some range defined by the image acquisition system. Due to technical and computational reasons the current range is usually 16 bits, while not all bits have to be used; special DICOM tags specify the exact layout. The following table lists typically used bit depths and value ranges for the modalities discussed in this work. Aside from these properties the data differ in their interpretation and thus need consideration for tone mapping alorithms. Computed tomography data, for example, has a fixed relation between the value at some position within the volume and the subject's density at that position. That is, the higher the measured volume the lower the radiographic density, and vice versa – a relation represented by the Hounsfield scale. In our case, transforming a CT signal I with some nonlinear global tone mapping function Φ would imply to interpret the data as if the Hounsfield function H had also been transformed, thus leading to the result I_H, that can be used for classification etc.:

Modality	Bit depth	Typical range
CT	12 (integer)	-1k...3k
MRI	10/12/16 (integer)	0...1-32k
fMRI	16 (integer/float)	-32k...32k
DTI	16/32 (float)	-32k...32k
PET	16 (integer/float)	0...32k

Table 2: *Typical properties of data in common modalities*

$$I_H = H(I) \quad \Longleftrightarrow \quad \Phi(I_H) = \Phi(H(I)) \tag{1}$$

In principle, only positron emission tomography (PET) also allows for a proportional relation between the measured intensity and the corresponding data (usually metabolic acitivity). All other modalities discussed here do not provide a direct mapping between the values and some scale, as even the values of different acquisitions can have different meaning. Things become even more complicated for diffusion tensor imaging (DTI) data, where not only scalar values per sample are to be interpreted: at least six entries for the partial derivatives in all directions are computed. Usually an additional channel specifies a "confidence" for the data at the current position. Thus, any function for mapping the data needs to be defined over six dimensions which is closely related to the topic of transfer functions that will be discussed further in section 4.

Based on these considerations it is clear that only global tone mapping provides a reasonable chance to maintain an interpretation of the data. Local operators, i.e., a non-uniform transformation of input data depending on local neighborhood etc., would change it in a way that interpreting or processing the data is practically impossible. Therefore, we concentrate on global operators that consider properties of the whole volume and can be adjusted by parameters similar to the current windowing technique. Although implementing such algorithms on graphics hardware faces the same computational complexity as CPU implementations, global tone mapping algorithms are especially suitable for GPUs due to their inherent parallelism. Thus, we expect a performance in the order of magnitude as shown before in table 1.

3.3 Tone mapping

As already described in sections 1 and 2, tone mapping is an important procedure in high dynamic range imaging. As explicated in the preceding section, the data in diagnostic visualizations is manually compressed by specifying a window of controllable width and center position. Although this linear interpolation between the minimum and maximum of the output device can also be regarded as tone mapping function, it cannot achieve proper, data-driven results automatically. In addition, the mapping will result in an information loss if the input range is larger than the output range – which is usually the case for medical data (see 3.2). In order to investigate the results of applying tone mapping algorithms from high dynamic range imaging to medical volume data while not sacrificing the real-time visualization of the system, we have decided to use global operators.

All of the following algorithms require some information about the data itself and some global user definable parameter. First, the so-called *background intensity* I_{avg} has to be estimated. Instead of simple averaging the intensities by computing the arithmetic mean, we have used the geometric average as suggested by Reinhard [Reinhard et al. 2005]:

$$I_{avg} = \exp\left(\frac{1}{N}\sum_{i=1}^{N}\log(I_i + \varepsilon)\right) \tag{2}$$

In addition, the unitless parameter α is known as the *key* and represents the overall light level of the data in an interval of $[0...1]$. Let L be the luminance if the input data, then α can be estimated according to [Reinhard et al. 2005] by: [2]

$$f = \frac{2\log L_{avg} - \log L_{min} - \log L_{max}}{\log L_{max} - \log L_{min}}, \quad \alpha = 0.18 \cdot 4^f \tag{3}$$

While α can also be controlled by the user, providing some reasonable default value is usually preferable. Some of the following algorithms that we have implemented introduce further parameters that will be discussed with the according description.

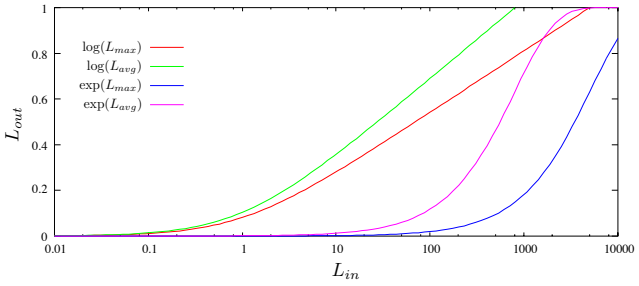


Figure 6: *Graphs for the default logarithmic/exponential tone mapping curves with varying parameters.*

3.3.1 Logarithmic and exponential scaling

Using the logarithm or an exponential mapping function is quite straightforward and has its background in Weber-Fechner's law considering the relationship between measured and perceived stimuli. However, these approaches only achieve reasonable results for medium ranged images, i.e., in our case CT or low-valued MRI data. First the logarithmic function: [3]

$$L_o(\vec{x}) = \frac{\log(1 + L_i(\vec{x}))}{\log(1 + L_{i_{max}})} \tag{4}$$

[2] Note that the logarithms used in the following equations are assumed to be to the base of 10, if not stated otherwise.

[3] For the equations in the following subsections we use the luminance function's input parameter as n-dimensional vector $\vec{x}$, where n denotes the dimension of the data ($n = 3$ for volumetric scalar data)

The exponential function in (5) maps input luminances to output luminances, where input values close to zero are mapped to zero, infinitely bright values are mapped to 1.0. This implies a renormalization because the output will never cover the full range available. Reinhard [Reinhard et al. 2005] reports that exchanging $L_{i_{max}}$ with $L_{i_{avg}}$ and vice versa yields a different effect, also shown in the corresponding graph in figure 6.

$$L_o(\vec{x}) = 1 - \exp\left(-\frac{L_i(\vec{x})}{L_{i_{avg}}}\right) \tag{5}$$

3.3.2 Extended logarithmic scaling

Following the logarithmic behaviour of the human visual system, Drago et al. [Drago et al. 2003] have further investigated in improvements of logarithmic functions. They proposed to adjust the logarithm's base with the input value and thus achieve a wider range of values to be reasonably mapped. As can be seen in the second term of the following equation, this base is interpolated within the range of 2 and 10:

$$L_o(\vec{x}) = \frac{L_{o_{max}} \cdot 0.01}{\log(1 + L_{i_{max}})} \cdot \frac{\log(1 + L_i(\vec{x}))}{\log\left(2 + 8\left(\frac{L_i(\vec{x})}{L_{i_{max}}}\right)^{\frac{\log(p)}{\log(0.5)}}\right)} \tag{6}$$

There are two parameters that can be specified by the user: the bias p controls the contrast, with larger values reducing the contrast. Also, the maximum output luminance $L_{o_{max}}$ (in cd/m^2) can be set, with a default value of 100.

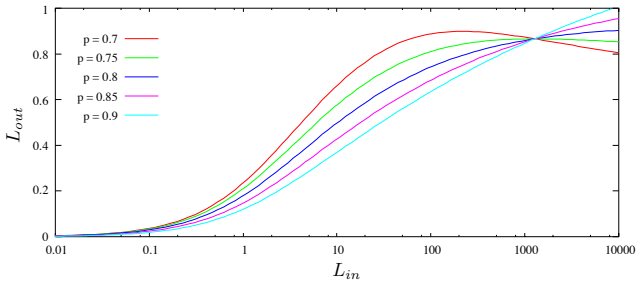


Figure 7: *Graphs for the tone mapping after [Drago et al. 2003] with varying parameters.*

3.3.3 Photoreceptor Model

While assuming that the human visual systems is of logarithmic nature is basically correct, Reinhard et al. [Reinhard and Devlin 2005] propose that this holds only for a certain range of values. As logarithms produce negative values and have no upper bound, they need to be modified for an adequate model. This leads to the following relation:

$$L_o(\vec{x}) = \frac{L_i(\vec{x})}{L_i(\vec{x}) + \sigma(L_{i_a}(\vec{x}))} \tag{7}$$

$$\sigma(L_{i_a}(\vec{x})) = (fL_{i_a}(\vec{x}))^m \tag{8}$$

$$k = \frac{L_{i_{max}} - L_{i_{avg}}}{L_{i_{max}} - L_{i_{min}}}, \quad m = 0.3 + 0.7k^{1.4} \tag{9}$$

In the equations (7) and (8), the term L_{i_a} denotes the adaptation level and can be set to $L_{i_{avg}}$ in our case, as no temporal or chromatic adaptation is needed. In addition, σ is often regarded as semisaturation constant. The following graphs illustrate the influence of the parameters k and f on the result:

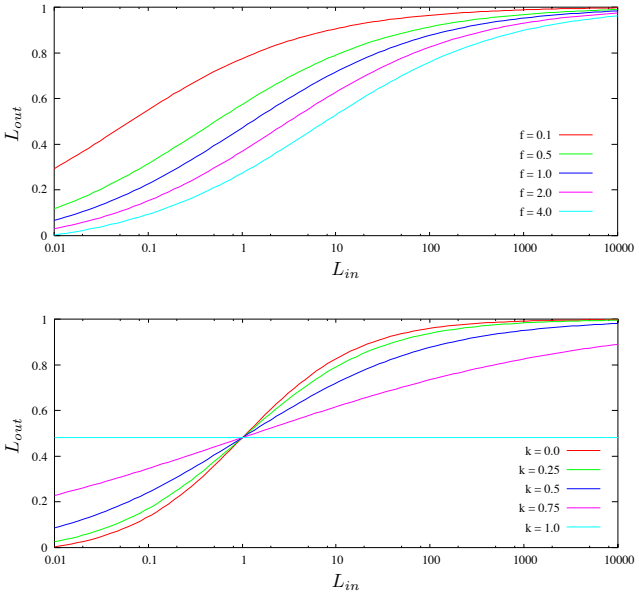


Figure 8: *Graphs for the tone mapping curves according to [Reinhard and Devlin 2005] with varying luminance parameter (top) and key parameter (bottom).*

4 Results and Discussion

We have used different kinds of datasets, focusing on modalities with a large dynamical range. Therefore, the data sets used in this paper are CT and PET data, with properties specified in the following table. As can be seen from the images, the rendering is in our case currently without using transfer functions. This is mainly due to the different ways how transfer functions can be used in the context of high dynamic range rendering and is subject to further investigation.

Data set	Resolution	Bit depth	min/max
CT backpack	$512 \times 512 \times 373$	12 (int)	0 / 4072
PET body	$168 \times 168 \times 715$	16 (int)	2 / 32767
PET head	$128 \times 128 \times 83$	16 (int)	0 / 32767

Table 3: *Properties of the data sets used in this paper*

4.1 Performance

As expected, the higher the dynamic range of values, the larger the improvement over linear mapping. MRI data sets, for example, usually provide values up to approximately 1k and thus do not benefit much from tone mapping. As can be seen in figure 9, the implemented algorithms result in quite different renderings, with Drago's algorithm being a good candidate also for other modalities. In addition, with only one parameter (see 3.3.2) this operator is very easy to use.
The algorithms have been implemented directly within the displaying shader programs. Without exhaustive optimization, the performance overhead is negligible for all renderings: even ray casting of the whole volume maintains real time performance on current commodity hardware. Adjusting the user parameters as well as the statistical data of the volume data is done via uniform variables that are transferred to the graphics hardware only when updated and thus do not impose a performance penalty.

4.2 Usability

In comparison to the standard windowing scheme, only little interaction is needed with our approach. While this automatic initialization of the visualization parameters is generally considered positive, medical staff emphasize to still be able to manually focus on specific ranges. The parameters for the tone mapping algorithms, however, are hard to relate with the traditional approach. Therefore, we propose to combine the two techniques by approximating the automatically determined tone mapping function linearly with adequate windowing parameters.
Also, for registration purposes the visualization of the whole range of values is considered to be useful. The more detailed structures in the tone mapped data (as in figure 1, for example) provide a better visual guidance when specifying landmarks in two modalities with different value ranges, e.g. PET and CT or MRI data.

5 Conclusion and future work

In this paper, we have shown that using the GPU is advantageous for improving the visualization of volume data of higher dynamic range than the output device. After reviewing several approaches to compress high dynamic range data, we have discussed their applicability to medical data. Global operators offer a good way of transforming the data while being computationally less expensive and inherently amenable to parallel architectures. On the other hand, local operators would introduce an uncertainty into the visual representation impeding diagnostic interpretation. Therefore, we have implemented some representative global operators in our GPU framework and discussed how to extend the method in different ways.

As the results are promising, we would like to further investigate in this topic. First, applying such operators to other, more specific image modalities like fMRI/DTI has not been addressed in detail yet. Especially the growing importance of functional imaging with data not simply corresponding to scalar properties such as luminance or density will lead to more complex algorithms. With the real time performance at hand, visual approaches such as rendering different settings as preview (so-called ”design galleries”) would improve both the handling of the algorithms’ parameters and multiple dimensions.
Another direction of research is the role of the transfer function in the context of high dynamic range imaging. Similar to pre-/post-classification in volume rendering, one can use the (HDR) transfer function with the original data and apply tone mapping procedures afterwards. Alternatively, the transfer function can be accessed by (low dynamic) values of the volume that has been tone mapped before. The implications are subject to further research, as well as its applicability to different modalities, especially for non-scalar types.

Acknowledgements

We would like to thank Guido Lorenz for his great support in the development of our framework, as well as all the students helping to extend Cascada’s possibilities. Also thanks to Cornela Massin and Stephan Palmer for inspiring parts of this work by their theses, and our co-operation partners at the hospitals for discussion, collaboration and providing data sets. Additional data sets are courtesy of OsiriX’s sample image website [Rosset et al. 2007].
The author would also like to thank many people at the Centre for image analysis of Uppsala University for providing a great working environment during the first development stages of Cascada. Finally, we would like to thank the reviewers for their feedback to improve this paper.

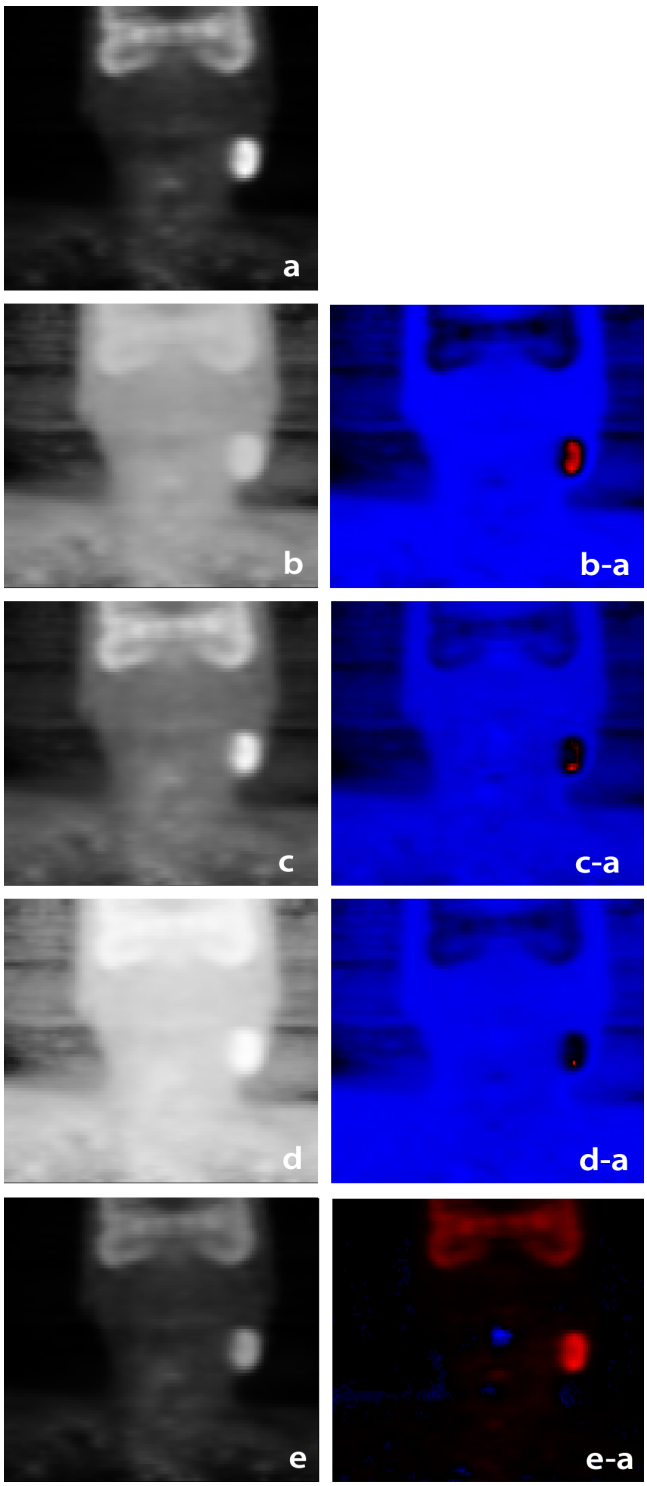


Figure 9: *Overview of all the tone mapping algorithms implemented, applied to a human head/neck PET scan (coronal view). The linear mapping (a) is compared to the operators by Reinhard (b), Drago (c) and the logarithmic (d) and exponential (e) methods. The right column shows the subtraction of the linear mapping, with blue colors denoting positive, red colors denoting negative values.*

References

ABRAM, G. D., AND WHITTED, T. 1990. Building Block Shaders. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 283–288.

BARTZ, D., SCHNAIDT, B., CERNIK, J., GAUCKLER, L., FISCHER, J., AND DEL RÍO, A. 2006. Volumetric High Dynamic Range Windowing for Better Data Representation. In *Afrigaph '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, ACM, New York, NY, USA, 137–144.

BUCK, I., FOLEY, T., HORN, D., SUGERMAN, J., FATAHALIAN, K., HOUSTON, M., AND HANRAHAN, P. 2004. Brook for GPUs: stream computing on graphics hardware. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 777–786.

COOK, R. L. 1984. Shade Trees. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 223–231.

DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH Conference Proceedings*, 369–378.

DRAGO, F., MYSZKOWSKI, K., ANNEN, T., AND CHIBA, N. 2003. Adaptive Logarithmic Mapping For Displaying High Contrast Scenes. In *Proc. of EUROGRAPHICS 2003*, Blackwell, Granada, Spain, P. Brunet and D. W. Fellner, Eds., vol. 22 of *Computer Graphics Forum*, 419–426.

FERNANDO, R., AND KILGARD, M. J. 2003. *The Cg Tutorial – The Definite Guide to Programmable Real-Time Graphics*. Addison-Wesley Professional.

GOODNIGHT, N., WANG, R., WOOLLEY, C., AND HUMPHREYS, G. 2003. Interactive Time-Dependent Tone Mapping Using Programmable Graphics Hardware. In *Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, 26–37.

HARRIS, M. J., BAXTER, W. V., SCHEUERMANN, T., AND LASTRA, A. 2003. Simulation of Cloud Dynamics on Graphics Hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 92–101.

HENSLEY, J., 2007. Close to the metal. Course 24 - GPGPU: General-Purpose Computation on Graphics Hardware. ACM SIGGRAPH, San Diego, CA., August.

LANGS, A., AND BIEDERMANN, M. 2007. Filtering Video Volumes Using the Graphics Hardware. In *SCIA*, Springer, B. K. Ersbøll and K. S. Pedersen, Eds., vol. 4522 of *Lecture Notes in Computer Science*, 878–887.

MCCOOL, M. D., QIN, Z., AND POPA, T. S. 2002. Shader Metaprogramming. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 57–68.

MCCOOL, M., TOIT, S. D., POPA, T., CHAN, B., AND MOULE, K. 2004. Shader Algebra. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 787–795.

MCGUIRE, M., STATHIS, G., PFISTER, H., AND KRISHNAMURTHI, S. 2006. Abstract Shade Trees. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, 79–86.

MEVIS, 2007. MeVisLab: A Development Environment for Medical Image Processing and Visualization. `http://www.mevislab.de`.

NVIDIA CORPORATION. 2007. *NVIDIA CUDA compute unified device architecture programming guide*, Jan. `http://developer.nvidia.com/cuda`.

OWENS, J. D., LUEBKE, D., GOVINDARAJU, N., HARRIS, M., KRÜGER, J., LEFOHN, A. E., AND PURCELL, T. J. 2007. A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum 26*, 1, 80–113.

REINHARD, E., AND DEVLIN, K. 2005. Dynamic Range Reduction Inspired by Photoreceptor Physiology. *IEEE Transactions on Visualization and Computer Graphics 11*, 1, 13–24.

REINHARD, E., STARK, M., SHIRLEY, P., AND FERWERDA, J. 2002. Photographic Tone Reproduction for Digital Images. *ACM Trans. Graph. 21*, 3, 267–276.

REINHARD, E., WARD, G., PATTANAIK, S., AND DEBEVEC, P. E. 2005. *High Dynamic Range Imaging – Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann, November.

ROSSET, A., ET AL., 2007. OsiriX Medical Imaging Software. `http://www.osirix-viewer.com`.

ROST, R. J. 2005. *OpenGL(R) Shading Language (2nd Edition)*. Addison-Wesley Professional.

VOLLRATH, J. E., WEISKOPF, D., AND ERTL, T. 2005. A Generic Software Framework for the GPU Volume Rendering Pipeline. In *Vision, Modeling, and Visualization VMV '05 Conference Proceedings,*, 391–398.

YUAN, X., NGUYEN, M. X., CHEN, B., AND PORTER, D. H. 2006. HDR VolVis: High Dynamic Range Volume Visualization. *IEEE Trans Vis Comput Graph 12*, 4, 433–45.

Real-time Generation of Plausible Surface Waves

Kristian Yrjölä and Thomas Larsson
Department of Computer Science and Electronics
Mälardalen University

Abstract

We present a fast and flexible algorithm for the simulation and rendering of ocean waves. The method is designed to support efficient view frustum culling and various simple wave effects such as choppy waves, capillary waves, wave refraction, round waves, and wave-land interaction, which makes the model suitable in, e.g., computer games. The waves are numerically robust, and the execution time of the generated waves can be controlled dynamically. Finally, experimental results illustrate the interactive performance and the visual quality of the generated waves.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: water, waves, animation, culling, graphics, simulation

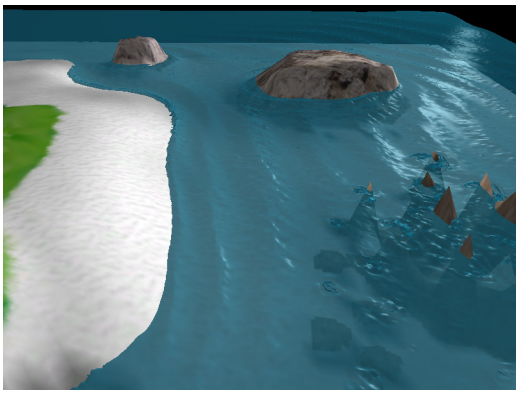


Figure 1: *An image of waves generated with the proposed method.*

1 Introduction

Simulation of realistic water waves is a computationally challenging problem [Semtner 2000; Hinsinger et al. 2002; Iglesias 2004]. There are many different types of waves, e.g., the tides, seismic tsunami waves, internal waves (waves below the water surface), object interaction waves, gravitational waves, and capillary waves [Peachey 1986]. The latter two are also called wind waves, and it is the generation of plausible wind waves in real-time that is in focus here.

Some previously proposed wave generating methods for computer graphics applications have been based on the summation of sine waves, such as Gerstner's and/or Biesel's wave models [Fournier and Reeves 1986; Cieutat et al. 2001; Hinsinger et al. 2002]. Additional details to the water surface can be added by using noise functions [Thon et al. 2000]. Wave refraction can be simulated with wave tracing [Ts'o and Barsky 1987; Gonzato and Sac 1997].

Other approaches rely on fast Fourier transforms [Tessendorf 2004; Hu et al. 2004], and solving the Navier Stokes equations for physically-based simulation of fluids [Kass 1991; Stam 1999; Foster and Fedkiw 2001; Mihalef et al. 2004]. Realistic simulation of fluids, however, can be extremely time consuming, and therefore inappropriate for interactive computer graphics applications. Nevertheless, Stam has proposed a fast and useful algorithm that solves the Navier–Stokes equations and produces complex fluid-like flows in real-time [Stam 1999; Stam 2003].

Several other methods are based on a combination of summing sine waves, spectral approaches, and other techniques [Thon et al. 2000; He et al. 2005]. Some approaches also use the modern programmability features of GPUs [Schneider and Westermann 2001; Isidoro et al. 2002; Kryachko 2005; Baboud and Décoret 2006]. We refer to the survey by Iglesias [2004] for a broader presentation of related work than the one given here.

Our solution is based on oriented Wave Train Boxes (WTBs), which are responsible for generating waves over spatially limited sea areas. The basic shapes of the waves are computed using Gerstner's equations [Tessendorf 2004], but by using several extensions and tracing wave rays inside the WTBs in the wind direction, simple forms of some natural wave effects are achieved, e.g., wave–land collisions, wave refraction, capillary waves, and round waves. In Figure 1, an example image of the generated waves is shown. The computational complexity of our approach is linear in the number of water surface points.

The main contribution of this paper is a simple and flexible wave model for the fast generation of plausible water surfaces with the following advantages: (1) The generated waves are frame-independent, which means the water surface is re-calculated from its rest position each frame. This is both memory friendly and numerically robust. (2) Wave trains are spatially limited, and mutually independent, which means that View Frustum Culling (VFC) can be used to speed-up the calculations. (3) The complexity of the water surfaces can be interactively controlled, since the computations are arranged in layers with the most basic features applied first. Furthermore, the water surface is generated as a combination of an arbitrary number of independently processed simple wave trains, so the most important can be selected according to a given time budget. (4) The model supports a simple and fast method for affecting the waves near land (wave refraction). (5) Land obstacles are detected, and in the water area behind them, the waves are canceled out. (6) The method produces plausible waves at interactive rates, see benchmarks in Section 3.

2 Wave model

A water surface is represented by a uniformly sampled height field which we call the water grid. For the generation of the waves, oriented WTBs are associated with the water grid. A WTB is used to handle the creation and simulation of a restricted rectangular area of waves. Each WTB handles a set with one or more wave trains traveling in roughly the same direction.

To create waves, a wind is associated with each WTB. Wind parameters are wind speed, direction vector, wind origin, life length, and area defined by fetch and width. When a wind starts, the size of the WTB grows during a build-up phase up to a maximum. Some properties of the wind directly define parts of the WTB, such as direction, width and origin point. The other properties help to build-up, sustain and in the end kill the WTB.

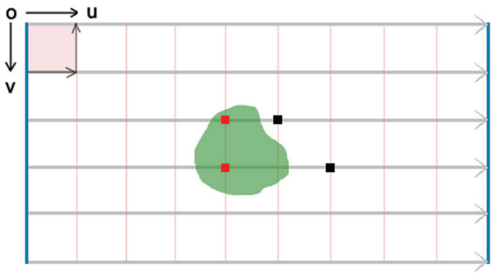


Figure 2: *A WTB with six wave rays. The internal coordinate system has its origin in the lower left corner when looking along u. Action points are registered around obstacles in order to change wave train parameters at those specific points.*

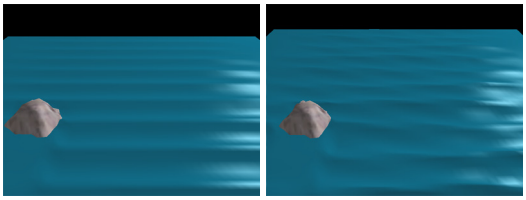


Figure 3: *Images generated from a WTB with a single wave train (left), and a WTB with three wave trains (right). As expected, more wave trains give a more irregular surface.*

There are one or more wave trains per WTB, each with its own direction, amplitude, wave length, and phase. The more wave trains used, the more complex waves are generated. Example waves resulting from a WTB with a single wave train and another with three wave trains are shown in Figure 3.

A WTB also has a set of wave rays in order to gain control over the wave train parameters locally. With this local control, the height of a wave can be increased as it approaches land, and waves behind an obstacle can be damped. However, when there are no land obstacles inside a WTB, it is possible to iterate over the water grid points in any order since each point's location is then calculated independently.

In Figure 2, the most important structural elements of a WTB are shown. The direction vectors u and v define the internal coordinate system of the WTB, with the origin always in the lower left corner looking along u. The lengths of these vectors are the actual uniform lengths between the water grid points. Thus, the WTB has as many wave rays as it spans over grid points in width. If something obstructs the surface, action points are registered as shown in Figure 2. An action point indicates a change of the wave train parameters will take place at that specific point.

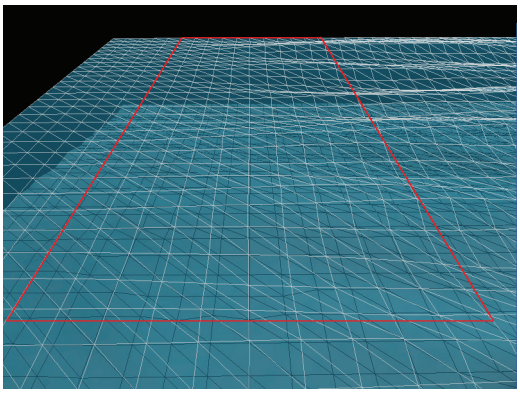


Figure 4: *Smoothening area along the side of a WTB.*

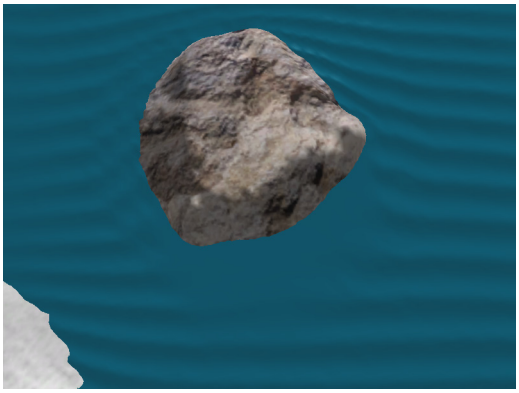


Figure 5: *Dampening of waves due to wave ray land collisions.*

The actual computation of the wave shape is based on Gerstner's wave model, which was suggested more than two hundred years ago in oceanography (see e.g. [Tessendorf 2004; Fournier and Reeves 1986]). This model is chosen because its inherent linear time complexity in the number of computed water surface points, but still, the computed wave shapes look plausible.

Waves are created by specifying a wind speed, h. The maximum wave height, H, of fully developed seas can then be approximated by

$$H = \alpha \frac{h^2}{g} \tag{1}$$

where g is the gravitational constant, and α is a dimensionless constant. This simple equation is the result of an early hypothesis suggesting a direct relation between wind speed and wave height. A reasonable approximation for fully developed wave heights can be computed by using $\alpha = 0.21$ [Resio et al. 1999][1].

The animation of the water grid points is handled by Gerstner's parametric wave equations. The goal is to make a point $\mathbf{p}_0 = (x_0, z_0)$ on the rest surface of the water (where $y_0 = 0$) to travel in a stationary circular orbit. However, by displacing the point at time t to

$$\mathbf{p} = \mathbf{p}_0 - \frac{\mathbf{k}}{k} A \sin(\mathbf{k} \cdot \mathbf{p}_0 - wt) \tag{2}$$

$$y = A \cos(\mathbf{k} \cdot \mathbf{p}_0 - wt) \tag{3}$$

the surface is perturbed into repeated trochoids, thereby also supporting the generation of so called choppy waves, depending on the relationship between k and A, where A is the amplitude. The wave vector, $\mathbf{k} = (k_x, k_z)$ is the horizontal direction in which the wave travels. The magnitude of $\mathbf{k}$ is related to the wave length λ, and is given by $k = 2\pi/\lambda$. The frequency w is related to k and the water depth D at $\mathbf{p}_0$ according to

$$w = \sqrt{gk \tanh(kD)} \tag{4}$$

This equation makes the waves slow down as the water depth decreases. However, when the water depth is large enough, we can

[1] Fournier and Reeves [1986] suggest another simple equation for determining the wave height given only the wind speed.

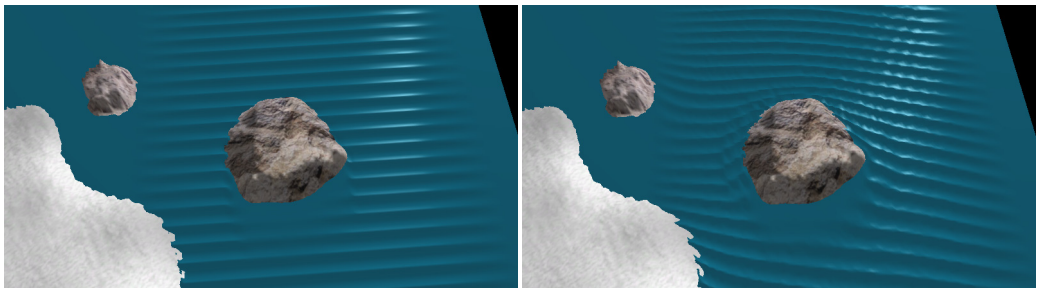


Figure 6: *Waves with no phase shift (left) and waves with a phase shift as a function of water depth (right).*

ignore the depth and simply use

$$w = \sqrt{gk} \tag{5}$$

Note that the equations 2 and 3 only create a wave with a very unrealistic regular shape. Therefore, to create a plausible wave shape, a set of N such sine waves are used and added together according to

$$\mathbf{p} = \mathbf{p}_0 - \sum_{i=0}^{N-1} \frac{\mathbf{k}_i}{k_i} A_i \sin(\mathbf{k}_i \cdot \mathbf{p}_0 - w_i t + \Phi_i) \tag{6}$$

$$y = \sum_{i=0}^{N-1} A_i \cos(\mathbf{k}_i \cdot \mathbf{p}_0 - w_i t + \Phi_i) \tag{7}$$

As can be seen, each sine wave has its own wave vector, k_i, amplitude, A_i, frequency w_i, and a phase shift Φ_i.

To avoid waves that are abruptly ended at the sides of the WTBs, a way to make the waves gradually disappearing into the water is needed. Therefore, each WTB has a smoothening area along its sides where the wave height is decreasing linearly, which is illustrated in Figure 4.

When the height of the water grid points inside the WTB is updated, the internal coordinate system is traversed along the wave rays. Since the rays are arbitrarily oriented, we have to detect which water grid points are closest. Some precautions must be taken to avoid neighboring rays updating the same grid points, or that no rays will access some grid points, resulting in visual artifacts. Therefore, for every internal grid point, the small quadrant area behind to the left is scanned for actual surface grid points, marked as a pink square in Figure 2. Because the internal grid of the WTB has the same distance between points as the water grid, albeit oriented, there can only be three possible outcomes for such a scan: zero, one or two points inside. The height is summed up for all wave trains at these points, and stored. If there is an action point, the parameters of the wave trains are adjusted. If the wave is approaching land and the depth is decreasing, the amplitude is increased slightly, but set to zero as land peaks above the surface. After the obstacle, if it is water behind it (from the wave direction point of view), the amplitude is increased again gradually to the full amplitude.

By computing the position of the water grid points along the wind direction, land collisions becomes trivial to detect, and the handling of waves hitting a land obstacle becomes extremely efficient. The land collision points can even be pre-computed as long as the WTBs are stationary. In Figure 5, this way of detecting land collisions and setting the wave amplitude to zero afterwards is shown. In nature, however, the broken waves would pass along the sides of the obstacle, grow along the crest and perhaps meet again depending on the size of the obstacle. This phenomenon is called diffraction.

Finally, note that some visual artifacts may arise during the wave animation near land when the amplitude is too high or the slope is low (like on a beach) because of land peeking through the bottom of the waves. One possible solution to this could be to adjust the parameters of the waves to make the trough of the wave stay above land. For example, the water level can be raised somewhat to simulate run-up, and decreasing the amplitude instead of increasing it can also help. Some visual artifacts may also arise when polygons of the water surface are coplanar with land polygons due to z-fighting. Increasing the depth buffer resolution can alleviate these artifacts, but can never remove the problem completely.

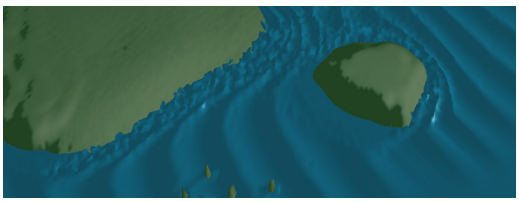


Figure 7: *Example of waves starting to queue up towards land.*

2.1 Wave effects

While updating the parts of the water grid covered by a WTB, several effects adding more details and variations can be applied. For example, a simple form of wave refraction can be achieved simply by shifting the phase of the waves locally depending on the water depth. In Figure 6, it is shown how the resulting waves look with and without this type of refraction. In contrast to this, the usual way to achieve wave refraction for Gerstner waves is to change the wave frequency according to Equation 4. However, when the frequency is changed, waves may easily start to queue up towards land as the animation proceeds according to Figure 7. This is because the speed is reduced to almost zero. Our approach keeps the propagation speed, and focuses on bending the waves.

In reality, there are also a great number of water waves that make up irregular patterns of waves with different heights. To model a seemingly irregular height variation, we can simply change the height value y along a wave in the v-direction of the WTBs by modifying the amplitude according to the formula

$$A' = A - a \sin(t + u) \sin(vf) \tag{8}$$

where a is scale factor, and f determines how often a higher wave will appear in the v-direction. Of course, some randomness can also be added to Equation 8. However, the resulting waves look more chaotic even without it, as can be seen in Figure 8.

There are also capillary waves giving rise to fine grained details on the water surface. Therefore, we add more details to the water

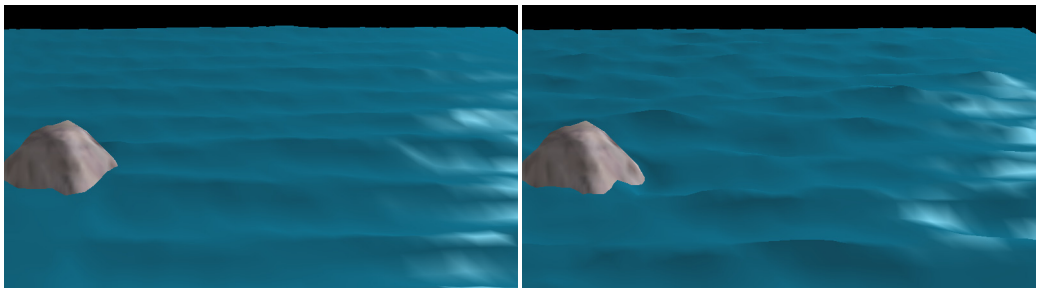


Figure 8: *Waves without the sideways height variation effect (left), and with it (right).*

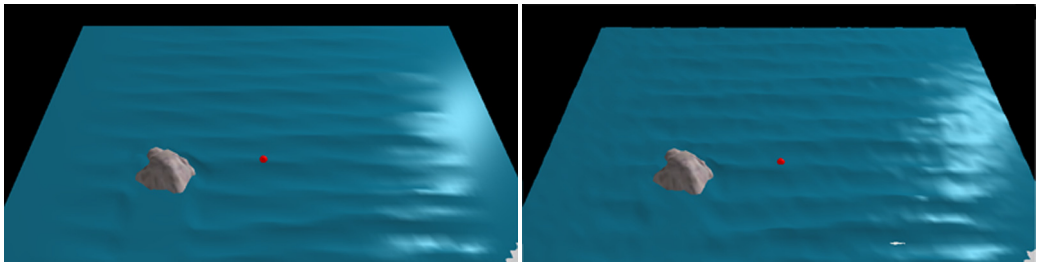


Figure 9: *A water surface without noise (left) and with noise (right).*

surface by applying a noise function adjusting the height of each water grid points slightly using the equation

$$y' = y + b\sin(rt) \tag{9}$$

where b is the noise amplitude, and $r \in [0, 1]$ is a random value. The effect of this noise function is shown in Figure 9. To avoid a completely flat water surface in areas where no wave trains are applied, the noise function must be applied directly to all visible water grid points, independently of where the WTBs are located.

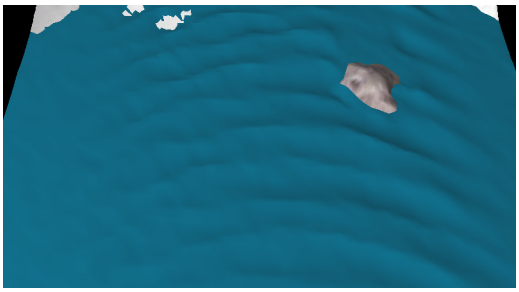


Figure 10: *Round waves resulting from waves spreading outwards from the water–wind contact area.*

Another effect that may arise can be referred to as round waves. When a wind blows over a surface with a limited width, the arising waves tend to spread outwards from the contact area. This effect can easily be incorporated into our model by a function shifting the phase of the waves in the v-direction of the WTB. For example, using the formula

$$\Phi = \frac{(v - 0.5V)^2}{V} \tag{10}$$

where V is the width of the WTB, the phase shift becomes symmetric around the center of the WTB giving rise to the round waves depicted in Figure 10.

2.2 View frustum culling

The concept of WTBs is well-suited for view frustum culling. Each WTB is a container for wave trains, and it defines a limited rectangular area where these wave trains affect the water grid points. For each WTB, we create an enclosing volume, an Oriented Bounding Box (OBB). The dimensions of the OBB are defined by the rectangular area of the WTB together with the maximum wave height, which is given by the sum of all the maximum heights of all involved wave trains.

The actual visibility check is made by extracting the planes of the view frustum each frame from the combined view and projection transformation matrices $\mathbf{A} = \mathbf{MP}$, which is done very efficiently by adding columns of $\mathbf{A}$ [Gribb and Hartmann 2001]. Then an OBB–plane overlap test is executed for each plane of the frustum, which can be computed very efficiently [Akenine-Möller and Haines 2002].

However, we need to take precaution to avoid potential false culling of overlapping WTBs. When two or more WTBs overlap, the maximum wave height inside the overlap area is the sum of the maximum wave height for each involved WTB. Therefore, when we determine the height of the OBB, we take all overlapping WTBs into account. Note that this computation can be done as a pre-process, if the WTBs are placed at fixed locations in the virtual environment.

2.3 Interactive control

To be able to interactively control the frame rate, our approach admits generating the waves using a computationally layered approach, where the most fundamental and important features are applied first. We start with conservative view frustum culling of the WTBs. Then, for each remaining WTB, the positions of the affected water grid points are computed, as they are passed by the wave rays. Note that the actual computations of the positions of the water grid points are made with the currently enabled wave trains and the possible effects either turned on or off. Finally, the noise effect may be added to the water grid.

If time permits, it would also be possible to make an additional pass over all water grid points that fall inside any of the WTBs to add other special effects, such as foam or spray, by considering the roughness of the water around each water grid point. It would also be possible to add new WTBs on the fly. However, then the computation of overlapping WTBs, and land collision points, must also be done on the fly, which otherwise are pre-computations.

2.4 Computational complexity

The performance of the algorithm is adaptive to the actual number of water grid points computed per frame. Since a single water surface point is processed in constant time, the worst case for the computational complexity of the algorithm is given by $O(kn)$, where k is the number of WTBs, and n is the number of water grid points.

In the best case, the algorithm is in $O(k)$, which occurs when all WTBs, and thus all wave trains, are completely outside the view frustum, and when the (global) noise effect is turned off. Therefore, from a scene design point of view, it makes sense to define the size of neighboring WTBs with efficient VFC in mind.

3 Results

Our wave generation method has been implemented in C++ and we have evaluated the performance by using some benchmark scenarios. The test equipment was an AMD64 3000+ 2.0 GHz CPU, with 1 GB RAM and graphics hardware ATI9800 pro with 128 MB RAM. No optical effects were used in the lighting calculation except OpenGL's own fixed-function pipeline with one light source. All animation calculations are done on the CPU, and VFC is only done for WTBs and not for the land and water geometries themselves. Some captured animations are available showing our wave generation approach in action[2].

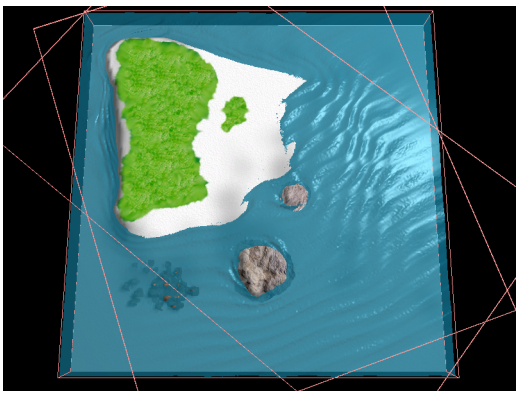


Figure 11: *Bird-eye view over the WTBs in the first scenario.*

LOD	64x64	128x128	256x256	512x512
WC	3.20 ms	12.2 ms	49.9 ms	199 ms
WR	0.80 ms	2.80 ms	10.8 ms	44.0 ms
FPS	>75	42	13	4

Table 1: *Wave Calculation (WC) and Water Rendering (WR) times in the first scenario.*

In the first scenario, there are two WTBs, both of them covering most of the water surface, see Figure 11. One of the WTBs includes one wave train, and the other two wave trains. The effects

[2] See http://www.idt.mdh.se/personal/tla/waves/

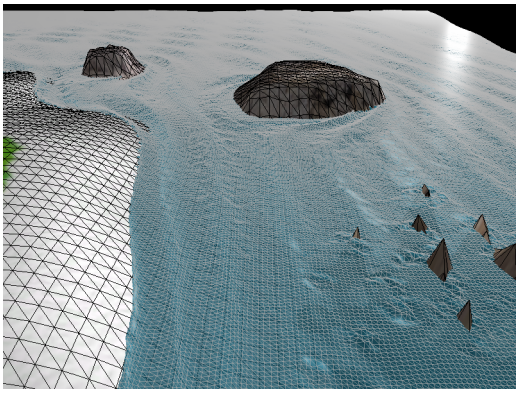


Figure 12: *Wireframe rendering of the waves in the first scenario with water grid LOD 512x512.*

used were basic Gerstner waves and depth-dependent wave refraction. The measurements were repeated for four different levels-of-detail (LODs) of the water grid, see Figure 12 for an example. The execution times are presented in Table 1. As can be seen, the time to generate the waves increases linearly with the number of grid points as expected.

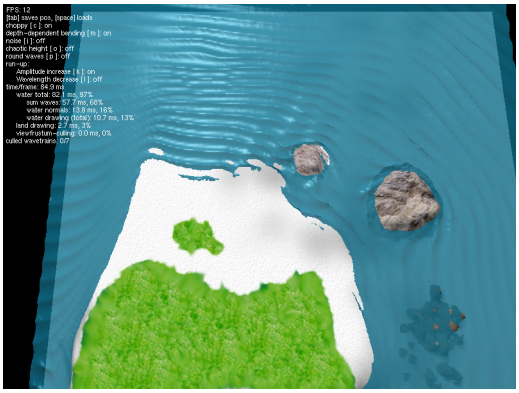


Figure 13: *Scenario 2 with seven WTBs inside the view frustum.*

In the second scenario, seven WTBs, with one wave train in each, were included to be able to examine how VFC affected the execution time. A fixed water grid of size 256x256 was used. Again, basic Gerstner waves and depth-dependent wave refraction were used. When all seven WTBs were visible (see Figure 13) the water calculation time was 57.7 ms. With none of the WTBs visible, the water calculation time decreased to 0.6 ms. In cases where one or more WTBs were culled, a speed-up of 3–10 ms per WTB was observed depending on their sizes. When the water grid size was increased to 512x512 in the same scenario, the water calculation time was 232.2 ms with all WTBs inside the view frustum, and the gain per culled WTB was between 20–40 ms.

Effect	256x256	512x512
Depth-dependent refr.	1.2 ms	6 ms
Noise	3 ms	13 ms
Irregular height	9.5 ms	39 ms
Round waves	0.5 ms	1.4 ms

Table 2: *The execution times of the tested wave effects for two LODs of the water grid.*

The second scenario was also used to measure the computation time for various parts of the algorithm. The measured parts were depth-dependent refraction, noise, irregular height, and round waves. The results from this experiment, with all seven WTBs inside the view frustum, are shown in Table 2.

4 Conclusion and future work

Even though realistic physically-based simulation of ocean waves is too complex for real-time graphics application, it is clear that simpler solutions, with a plausible look and feel, can be incorporated with success in, e.g., computer games or virtual reality applications. The key features which make plausible wind waves possible in our approach are the linear time complexity, flexible and independent wave effects, and VFC of WTBs. We expect that an implementation of our approach based on, e.g., the SIMD SSE instruction set and multiple core CPUs, would improve the performance by an order of magnitude making the computation of water surfaces of 1024x1024 grid points feasible in real-time. Some computations can also be moved to the GPU for further improvements.

Possible future work includes examining various approaches to add plausible foam, spray and breaking waves in an additional pass over the visible WTBs. Working with dynamically moving WTBs would also be interesting, e.g., to achieve waves generated by boats in motion. Another possibility would be to further accelerate the wave generation process by improving the VFC. For example, this includes examining ways to efficiently clip WTBs only partly inside the frustum.

References

AKENINE-MÖLLER, T., AND HAINES, E. 2002. *Real-Time Rendering (2nd Edition)*. A K Peters, Ltd.

BABOUD, L., AND DÉCORET, X. 2006. Realistic water volumes in real-time. In *Eurographics Workshop on Natural Phenomena*, Eurographics.

CIEUTAT, J. M., GONZATO, J. C., AND GUITTON, P. 2001. A new efficient wave model for maritime training simulator. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics*, IEEE Computer Society, Washington, DC, USA, 202.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 23–30.

FOURNIER, A., AND REEVES, W. T. 1986. A simple model of ocean waves. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 75–84.

GONZATO, J.-C., AND SAC, B. L. 1997. A phenomenological model of coastal scenes based on physical considerations. In *8th Eurographics Workshop on Computer Animation and Simulation*, Springer-Verlag, Berlin, Heidelberg, Germany, 137–148.

GRIBB, G., AND HARTMANN, K., 2001. Fast extraction of viewing frustum planes from the world-view-projection matrix.

HE, H., LIU, H., ZENG, F., AND YANG, G. 2005. A way to real-time ocean wave simulation. In *Proceedings of the Computer Graphics, Imaging and Vision: New Trends (CGIV'05)*, IEEE Computer Society, Washington, DC, USA, 409–415.

HINSINGER, D., NEYRET, F., AND CANI, M.-P. 2002. Interactive animation of ocean waves. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 161–166.

HU, Y., VELHO, L., TONG, X., GUO, B., AND SHUM, H. 2004. Realistic, real-time rendering of ocean waves. *Computer Animation and Virtual Worlds. Special Issue on Game Technologies.*.

IGLESIAS, A. 2004. Computer graphics for water modeling and rendering: a survey. *Future Generation Computer Systems 20*, 8, 1355–1374.

ISIDORO, J., VLACHOS, A., AND BRENNAN, C. 2002. Rendering ocean water. In *ShaderX: Vertex and Pixel Shader Tips and Tricks*. Wordware Publishing.

KASS, M. 1991. Height-field fluids for computer graphics. In *WSC '91: Proceedings of the 23rd conference on Winter simulation*, IEEE Computer Society, Washington, DC, USA, 1194–1198.

KRYACHKO, Y. 2005. Using vertex texture displacement for realistic water rendering. In *GPU Gems 2, Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley, ch. 18, 283–294.

MIHALEF, V., METAXAS, D., AND SUSSMAN, M. 2004. Animation and control of breaking waves. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 315–324.

PEACHEY, D. R. 1986. Modeling waves and surf. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 65–74.

RESIO, D. T., SWAIL, V. R., JENSEN, R. E., AND CARDONE, V. J. 1999. Wind speed scaling in fully developed seas. *Journal Of Physical Oceanography 29*, 1801–1811.

SCHNEIDER, J., AND WESTERMANN, R. 2001. Towards real-time visual simulation of water surfaces. In *VMV*, 211–218.

SEMTNER, A. 2000. Ocean and climate modeling. *Communications of the ACM 43*, 4, 80–89.

STAM, J. 1999. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128.

STAM, J. 2003. Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference*.

TESSENDORF, J. 2004. Simulating ocean water. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, ACM Press, New York, NY, USA.

THON, S., DISCHLER, J.-M., AND GHAZANFARPOUR, D. 2000. Ocean waves synthesis using a spectrum-based turbulence function. In *CGI '00: Proceedings of the International Conference on Computer Graphics*, IEEE Computer Society, Washington, DC, USA, 65.

TS'O, P. Y., AND BARSKY, B. A. 1987. Modeling and rendering waves: wave-tracing using beta-splines and reflective and refractive texture mapping. *ACM Trans. Graph. 6*, 3, 191–214.

Clustering Geometric Data Streams

Jiří Skála*
Ivana Kolingerová†
University of West Bohemia

Abstract

Using recent knowledge in data stream clustering we present a modified approach to the facility location problem in the context of geometric data streams. We give insight to the existing algorithm from a less mathematical point of view, focusing on understanding and practical use, namely by computer graphics experts. We propose a modification of the original data stream k-median clustering to solve facility location which is the case when we a priori do not know the number of clusters in the input data. Like the original, the modified version is capable of processing millions of points while using rather small amount of memory. Based on our experiments with clustering geometric data we present suggestions on how to set processing parameters. We also describe how the algorithm handles various distributions of input data within the stream. These findings may be applied back to the original algorithm.

CR Categories: I.5.3 [Computing Methodologies]: Pattern Recognition—Clustering; I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling

Keywords: data stream, clustering, facility location, geometric data

1 Introduction

Data stream algorithms have been extensively studied in connection with databases and network statistics. However, there is not much research dedicated to geometric data streams. Since geometric models are growing larger and larger like those from Stanford's 3D Scanning Repository [Stanford 2007], data stream approach is becoming essential to process such models. By clustering we can significantly reduce the amount of data. Clusters can be then used to create a multiresolution model. Data stream clustering is done in a hierarchical way which gives a possibility to control the use of memory. The algorithm runs with just several megabytes of memory while processing millions of points.

In our paper we propose a method for clustering geometric data in a streaming fashion. Assuming we do not know how many clusters there are, we think of the clustering as a facility location problem. We present a modified k-median algorithm to solve the facility location. Data stream processing is performed in blocks. Semi-results from blocks are processed at higher levels. This leads to a hierarchy which offers a possibility to build a multiresolution model. Unlike many algorithms for large geometric data, our method can process unordered points without any pre-processing. Based on our experiments we give suggestions of algorithm parameter settings and describe how the method works with various input data.

Section 2 briefly introduces problem background. Section 3 presents current state of the art in clustering and data stream algorithms. Sections 4 and 5 describe the method in detail. Our modifications and improvements to the current algorithm are described in Section 6. Section 7 presents the experimental evaluation. In Section 8 we give a conclusion and suggest future work.

2 Background

2.1 Clustering

Clustering stands for a wide range of problems. It concerns grouping similar elements together to form clusters. Euclidean distance is most often used as a measure of similarity. Perhaps the most common is the k-centres or k-means clustering. In this case we divide the elements (also referred to as points) into exactly k clusters so that the sum of distances from each point to the corresponding cluster centre is minimized. Another formulation of the problem is the k-median or k-medoid clustering where we search for cluster centres only among input points.

However, not always we know the number of clusters k. This problem is known as the facility location. A facility can be understood as a cluster centre. Given a set of points we choose some of them and open a facility there. All other points are connected to the closest facility. The problem is then to determine at which points a facility should be opened.

Since we want to minimize the distance between each point and its facility, it would be best to open a facility everywhere. But that is not what we want. That is why we introduce a facility cost - a penalty for opening every facility. Since we assume no differences between facilities, we set the same cost for all of them. Then we have the sum of distances between points and facilities on one hand, and the sum of facility costs on the other hand. Facility cost determines the balance between the cluster size and the number of clusters. This leads to an overall clustering cost C which is the sum of expenses for opening facilities plus the sum of distances from points to their facilities.

$$C = k \cdot fc + \sum_{i=0}^{N-1} \parallel c_i - f_{ci} \parallel \quad (1)$$

where k is the number of open facilities, fc is the facility cost (the same for all facilities), N is the number of points and $\parallel c_i - f_{ci} \parallel$ denotes the distance of point c_i to its facility. Those points where a facility is opened are supposed to be assigned to themselves thus having zero distance from their facility.

2.2 Data Stream

Data stream is a sequenced set of data that can only be viewed in order; there is no random access possible. Moreover, the set is supposed to be too large to fit into main memory. The data may

*e-mail: jskala@kiv.zcu.cz
†e-mail: kolinger@kiv.zcu.cz ; Work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Centre for Computer Graphics).

come online in some time intervals or even the whole set may be stored but in some slow external memory. So the data stream must be processed in pieces and during one or very few linear scans.

3 State of the art

Clustering has been studied in many areas from many points of view. An overview of clustering techniques may be found in [Jain et al. 1999].

Several approaches exist for the facility location; a nice overview can be found in [Shmoys 2000]. A fundamental approach is based on linear programming relaxation [Chudak and Shmoys 2004; Charikar and Guha 1999]. The facility location problem is formulated as a linear programming (LP). The LP relaxation is solved in polynomial time and gives an approximate solution to the original problem. There is a related method also based on LP [Charikar and Guha 1999; Chudak and Shmoys 2004]. It uses a primal-dual scheme. A dual linear program is solved which gives a solution to the primal problem.

Another approach uses a technique of local search [Charikar and Guha 1999; Korupolu et al. 1998]. A coarse initial solution is iteratively improved by the local search. In each step a point is chosen at random and it is determined whether it is profitable to open a facility there. If so, nearby points are reassigned to this new facility. If there are facilities with a low number of points, these facilities are closed and points reassigned elsewhere.

There are methods for clustering large databases. CLARANS [Ng and Han 1994] views the problem of finding cluster centres as a graph searching problem. Each set of centres is interpreted as a graph node. By replacing one of the centres we get to a neighbouring node. The task is to get to a node with the minimal clustering cost. This is done by traversing the graph. To limit the computational complexity, CLARANS does a randomized search, i.e. it inspects just a random sample of neighbours for each node. BIRCH [Zhang et al. 1996] uses another approach. For each cluster it keeps a Clustering Feature - a vector of summary information about the cluster. Clustering Features are organized in a CF tree. Leaf nodes represent particular clusters. Higher nodes represent a cluster formed by all the children subclusters. CURE [Guha et al. 1998] uses a hierarchical clustering where small clusters are merged into larger ones. CURE keeps a well scattered sample of points for each cluster. It is a compromise between all-point and centroid-based cluster representation. Samples are then used to identify nearby clusters that should be merged together. Nevertheless, forenamed methods do not perform true data stream processing. They require the whole data set to be in the main memory.

[Muthukrishnan 2003] gives a nice overview of data stream algorithms. Data stream clustering is extensively studied by Guha et al. [Guha et al. 2000; O'Callaghan et al. 2002; Guha et al. 2003]. Using a hierarchical approach based on local search they solve the k-median problem. Data stream processing is performed in blocks. Semi-results from each block are maintained as a higher level stream which is continuously reprocessed in the same way, forming higher levels. [Meyerson 2001] deals with online facility location. For each newly arrived point the algorithm measures the distance to the nearest existing facility. It is then decided whether to connect the point to that facility or whether to open a new facility at the point. Charikar et al. propose algorithms that can handle outlier points in input data [Charikar et al. 2001]. The main idea is that some small fraction of points (outliers) may be left unassigned to any cluster. A penalty is assigned to every outlier point and the sum of penalties is added to the overall clustering cost (for the facility location it is Equation 1).

Stream processing of geometric data is extensively studied by Isenburg et al. Their research ranges from streaming formats for polygon meshes [Isenburg and Lindstrom 2005], streaming compression of geometric models [Isenburg et al. 2006a], to streaming computation of Delaunay triangulation [Isenburg et al. 2006b]. Other related works include polygonal models simplification [Lindstrom 2000] or approximate Voronoi diagrams [Sharifzadeh and Shahabi 2004]. Stream processing of points is addressed in [Pajarola 2005]. This work introduces so called stream operators which are applied while sweeping sorted data. Perhaps the first use of clustering for multiresolution models can be found in [Rossignac and Borrel 1993]. It uses a simple clustering to create approximations of 3D polyhedra for rendering complex scenes. More recent research in clustering geometric data streams is described in [Frahling and Sohler 2005] which is concerned with dynamic geometric data streams. In such a case we have a set of points and the stream consists of insert/delete operations among these points.

4 The clustering algorithm

As stated in Section 1 we will be concerned with the facility location problem. Our solution employs the Local Search algorithm proposed by Charikar and Guha [Charikar and Guha 1999]. First an approximate initial solution is generated. It is then iteratively refined by a series of local search improvements.

First suppose we already have some initial solution. Local search improvements can start. We take a point at random and we ask a question: What if we open a facility here? Would it be beneficial? First we will have to pay for opening the facility (if it is not already open). We then inspect all other points and compare the distance to their current facility with the distance to the new facility candidate. If the candidate lies closer, the point is reassigned to it, sparing some connection expenses. To limit computational complexity, any point can be reassigned only to the facility candidate. The time complexity of this first phase of one local search step is $\mathcal{O}(N)$, where N is the number of all points (we compute the distance of the facility candidate to all other points).

After that some facilities may contain just a few points. If we reassigned all of these remaining points (even if the new facility is farther), we would be able to close their old facility and spare the facility cost for it. But we must be careful whether the facility cost spared will overweight the expenses for reassigning points to a farther facility. This second phase has $\mathcal{O}(N)$ time complexity too. It could be computed simultaneously with the first phase.

So for some point p taken at random we determine whether it would be profitable to open a facility there. This is expressed by the *gain* function. Let us first define a *distance spare* ds_i as the distance we spare by reassigning point c_i to p. It is the difference between distances to the current facility and to the facility candidate p. If the difference is negative (current facility lies closer then p) we set $ds_i = 0$. Next we define a *close spare* cs_j as a cost we can spare by closing facility f_j. It is the facility cost minus expenses for reassigning all points from f_i to p. Again if cs_j is negative (we cannot spare anything) we set $cs_j = 0$. The gain function is then computed according to Formula 2

$$gain(p) = -fc + \sum_{i=0}^{N-1} ds_i + \sum_{j=0}^{M-1} cs_j \qquad (2)$$

where fc is the facility cost (zero if there is a facility already open at p), N is the number of all points and M is the current number of facilities. As stated before, computing function gain takes $\mathcal{O}(N)$ time.

Now to the initial solution. A very coarse one is sufficient, since local search will improve it quickly. All facilities have the same cost, so we use a simple algorithm proposed in [Meyerson 2001]. Points are taken in random order. At the first one a facility is always created. At every other point a facility is opened with probability d/fc, where d is the distance of the current point to the closest facility and fc is the facility cost. If $d/fc > 1$ we set the probability to 1.

The described local search technique is repeated $N \log N$ times. The number of iterations is derived in [Charikar and Guha 1999]. The initial solution can be generated in $\mathcal{O}(N^2)$ time. Function gain has $\mathcal{O}(N)$ time complexity and is evaluated $N \log N$ times. So the complete clustering algorithm has an overall time complexity of $\mathcal{O}(N^2 \log N)$.

5 Clustering a data stream

For clustering a data stream we use a hierarchical approach proposed in [Guha et al. 2000]. The algorithm inputs a block of points from the data stream and performs a clustering on it. Resulting facilities are given a weight according to the number of points assigned to them. Thus clusters containing more points have more importance. Weighted cluster centres are then passed to a higher level. Remaining points are discarded and the algorithm proceeds with another block from the data stream.

Facilities at higher levels are treated as weighted points and are also processed in blocks. When enough points gather to fill up an entire block, they are clustered again. This time the distance of a point to its facility is multiplied by the point weight. Resulting cluster centres are given a weight equal to the sum of weights of points assigned to them. Weighted facilities are again passed to a higher level. Figure 1 illustrates the hierarchical processing. Black dots indicate cluster centres in particular blocks. Blocks in the data stream are delimited by bold lines.

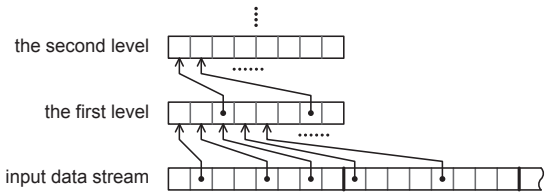


Figure 1: Hierarchical clustering.

The number of levels l required to process the entire data stream can be computed as

$$l = \frac{\log(N/m)}{\log(m/k)} \tag{3}$$

where N is the number of all points, m is the block size and k is the average number of clusters in one block. The equation was presented in [Guha et al. 2000] without any further explanation. Let us show how it can be derived. At the zero level the data stream is divided into N/m blocks. These blocks will be clustered, resulting in $N/m \cdot k$ facilities divided into $N/m \cdot k/m$ first-level blocks. The situation repeats at higher levels until resulting l-level facilities fit into a single block. We can write this as

$$N/m \cdot k/m \cdot k/m \cdot \ldots \cdot k/m = 1 \tag{4}$$

where k/m repeats l-times. After rearrangement

$$N/m = m/k \cdot m/k \cdot \ldots \cdot m/k = (m/k)^l \tag{5}$$

Taking a logarithm of Equation 5 we get

$$l = \log_{m/k}(N/m) = \frac{\log(N/m)}{\log(m/k)} \tag{6}$$

6 Our modifications and improvements

6.1 Making Local Search More Local

We may speed up the evaluation of function gain by limiting the number of points that need to be inspected. Given a facility cost fc, any point can be connected to a facility at most fc far away. Otherwise it is cheaper to open a new facility at that point. Let us define the *influence area* of facility f to be the circle with centre f and radius fc. All points connected to f must then lie within its influence area.

When computing the gain function we can inspect just those points whose distance from the facility candidate is at most fc. How do we find them? Any cluster can contain just points lying within radius fc from the cluster centre. So when we take all facilities in a $2 \cdot fc$ radius around the facility candidate, and examine all points connected to those facilities, we can be sure that we inspected all necessary points.

The situation is illustrated in Figure 2. Facilities are shown as dots with an influence area drawn as dotted circle. The facility candidate is designated black. We need to inspect all points that may lie within its influence area IA. Such points can only belong to facilities whose influence area overlaps with IA. Such facilities (shown as diamonds) lie within the dashed circle with radius $2 \cdot fc$.

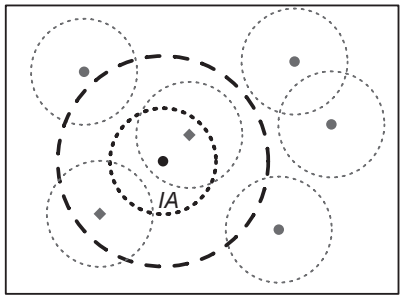


Figure 2: Finding points that may lie within the facility candidate's influence area.

For weighted points the situation is a bit more complicated because distances are multiplied by point weights. So a point with a small weight can be connected to a distant facility, while a point with a high weight should be connected to a nearby facility. But the above idea can still be used. For each facility we find the point with minimal weight w_{min}. The influence area radius is then fc/w_{min}. So we need to inspect all points assigned to facilities lying at most $2 \cdot fc/w_{min}$ away from the facility candidate. Note that w_{min} could be different for each facility.

6.2 From k-median to Facility Location

Cited papers deal with the k-median problem. They compute facility location repeatedly and using binary search they find such facility cost that yields exactly k clusters. Our modification computes the clustering just once. Solving the facility location itself seems to be just part of the original method, but it is not so simple. Since we have no k we must choose a suitable facility cost to get natural

clusters. We suggest setting the facility cost for each block of data equal to the diagonal of bounding box. This is discussed in detail in Section 7.1.

Another problem brings clustering at higher levels. Points have weights so all distances are multiplied by some (possibly large) numbers. If we perform the clustering as usual, a facility would be opened at almost every point because weights make them several times farther from each other. We could increase the facility cost but point weights will grow higher with increasing level and we may encounter numerical problems. Instead of scaling the facility cost we decided to introduce weight normalization.

Points at level zero have unit weight. We would like to keep weights around one also at higher levels. Therefore we divide all weights by their average. The average of new weights will be one, exactly as we wanted. It is important to do the normalization of all the points in a block at the same time. That means not earlier then the block is full. Normalizing weights right after clustering a lower level block (before passing points to higher level) is wrong. Each block may have a different number of clusters so the average weight may also vary. Thus points from different blocks would not be normalized equally.

7 Experiments and results

It this section we present experiments made with clustering geometric data. We discuss how the result depends on parameter settings and on the distribution of points in the stream. We give recommendations on how to set parameters to get a good clustering.

The algorithm was implemented in C# 2.0. Experiments were done on Intel Pentium 4 3.2 GHz with 2 GB RAM and SATA HDD, running Windows XP Professional. All measured times include I/O operations. Memory requirements were measured using the Performance Monitor in Windows XP.

7.1 Setting the Facility Cost

The facility cost determines how strongly the data will be clustered. High setting means an aggressive clustering resulting in a lower number of large clusters. Low setting will cluster just moderately producing many smaller clusters.

Some experiments are usually needed to find a facility cost that best fits your needs. It is a counterbalance to point distances. We need a small facility cost for clustering points in a unit square, and a high one for points in $[0; 10^6]$ interval. To avoid time consuming normalization, the facility cost must be derived from the range of point coordinates. Coordinate magnitude does not matter because we only care about point distances. We suggest setting the facility cost for each block equal to the diagonal of bounding box. It mostly produces good results. You can double the facility cost for a stronger clustering or divide it by two (or even by four) to get a moderate clustering.

Of course each block may have a different bounding box. So the clustering will be performed in each block with a different facility cost. But this is not a problem since we use weights. For a low facility cost we get many facilities with a low weight. High facility cost produces few facilities with big weight. Figure 3 shows an example of clustering with a facility cost set to double and half the diagonal respectively. Numbers show facility weights (before normalization).

Remember that a strong clustering reduces the amount of data faster. The algorithm may then run with fewer clustering levels, having lower memory requirements and shorter execution time. Indeed, the opposite holds for a moderate clustering.

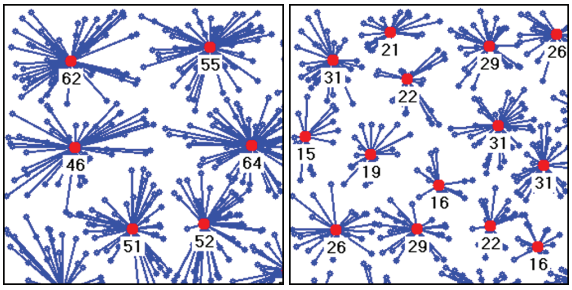


Figure 3: Clustering with a double facility cost (left) and a half facility cost (right). Figures cropped.

7.2 Input Point Distribution

Most authors concerned with large geometric data often rely on that input data will be more or less ordered. Our algorithm can handle unordered data as well.

If points arrive in order the algorithm processes them successively cluster by cluster. Transitions between clusters are generally handled correctly. The situation is illustrated in Figure 4. The dataset contains 2200 points in four groups (550 points each). It was processed with facility cost set equal to the bounding box diagonal, block size 750. Frames a)–c) show three blocks at level zero. Resulting facilities are passed to a block at the first level shown in Frame d).

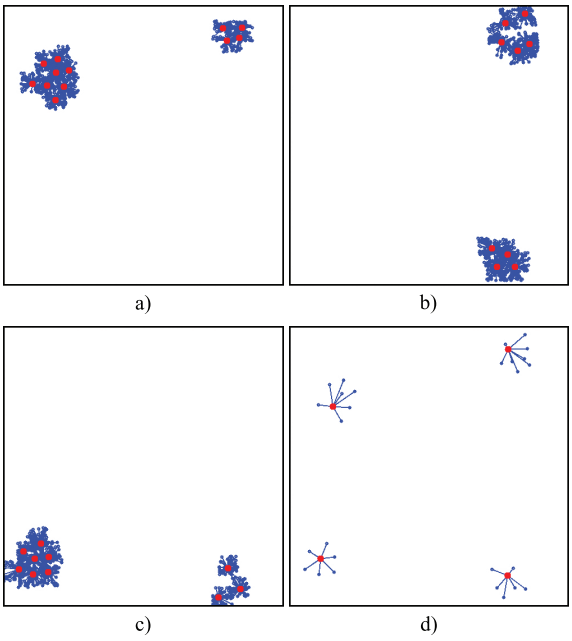


Figure 4: Clustering ordered data.

If points are scattered in the stream, and they come in a rather random order, it does not mean any trouble. The algorithm will process several points from different clusters at once. These points form something like cluster fragments which will be merged at higher levels. You can see an example in Figure 5. This is the same data

as in Figure 4, they were just shuffled. Processing parameters were also the same. Frames a)–c) show three blocks at level zero. Resulting facilities are passed to a block at the first level which is shown in Frame d).

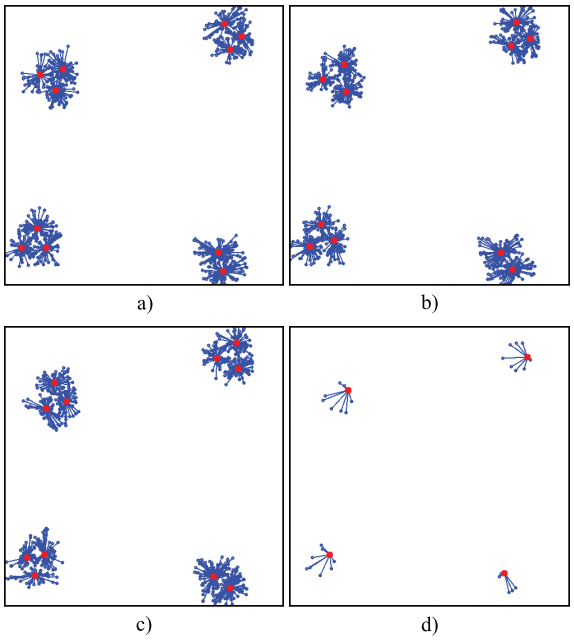


Figure 5: Clustering unordered data.

7.3 The Block Size

The clustering also depends on the size of block in which input data are processed. Block size basically affects execution time, the amount of memory used and also the clustering result.

Let N be the number of all points, m be the block size and k be the average number of clusters in each block. The number of all blocks at all levels will be

$$\begin{aligned} & N/m + N/m \cdot k/m + N/m \cdot k/m \cdot k/m + \ldots = \\ &= N/m \cdot [1 + k/m + (k/m)^2 + \ldots] = \\ &= N/m \cdot 1/(1 - k/m) = N/m \cdot m/(m - k) = \\ &= N/(m - k) \end{aligned} \tag{7}$$

As proved in [Charikar and Guha 1999], $m \log m$ local search iterations are necessary for each block. So the total number of iterations over all blocks will be

$$N/(m - k) \cdot m \log m \tag{8}$$

Because k is proportional to m, we can write

$$N/(m - c_1 \cdot m) \cdot m \log m = c_2 \cdot N \cdot \log m \tag{9}$$

where c_1, c_2 are some constants. So by decreasing block size m, the number of iterations necessary to process the whole data set also decreases.

Lowering the block size also decreases memory requirements. Let l be the number of all levels. The amount of memory required is proportional to

$$m \cdot l = m \cdot \frac{\log(N/m)}{\log(m/k)} \tag{10}$$

Since m/k can be considered constant, we can write

$$m \cdot l \approx m \cdot \log(N/m) \tag{11}$$

whereas $N \gg m$. Figure 6 shows a graph plot of Equation 11 for $N = 10^6$ (the black line). In practice we must use an integer number of levels (rounded up). This is shown as the grey line. You can see spikes where the number of levels changes.

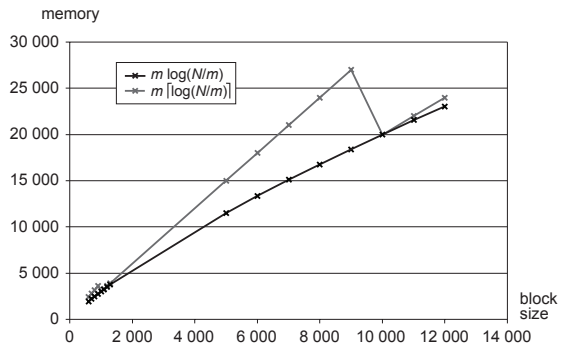


Figure 6: Graph plot of memory requirements.

It would seem that it is best to process data in very small blocks, but there is a drawback. When processing points in distinct blocks the result is an approximation of an ideal clustering. Of course the smaller the block, the worse the approximation. See [Guha et al. 2003] for details. According to our experiments, when varying block size, the clustering also varies somewhat but it still looks well. The major difference is that clustering in small blocks produces higher number of smaller clusters. Table 1 summarizes our experiments with the Lucy model [Stanford 2007], 10 072 906 vertices. The facility cost was set equal to the bounding box diagonal.

Table 1: The influence of block size on the clustering.

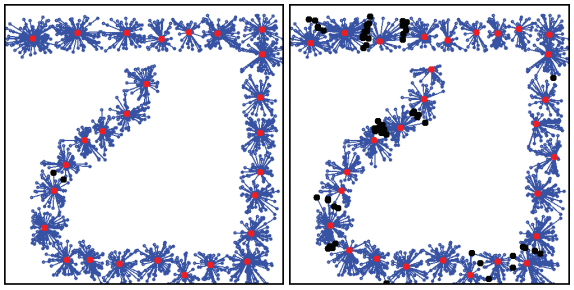
block size [points]	time [h:m]	memory [MB]	number of clusters at particular levels	
			level 1	level 2
1250	1:01	4.44	29 047	404
2500	1:58	3.97	28 400	366
5000	3:51	4.09	28 298	337
7500	5:43	5.27	28 270	336
10000	7:36	5.78	28 062	303

7.4 The Number of Iterations

[Charikar and Guha 1999] proved that $\mathcal{O}(m \log m)$ local search iterations are necessary for a constant factor approximation to the facility location. If we use large blocks, running time grows unpleasantly. We have made experiments with the number of iterations and it seems that it can be reduced significantly without major impact on results. Only about $0.1m$ iterations were necessary for uniformly distributed data. Data with obvious clusters required even less iterations.

You can see examples of clustering in Figure 7. The data set contains 1640 points. They were processed in a single block with facility cost equal to the bounding box diagonal. Black dots indicate points assigned to a different facility than to the closest one. It can be used as an approximate measure of error. But remember that it takes into account only the current set of facilities. No black dots mean optimal assignment to *currently open* facilities. With a different set of facilities, the clustering may be better.

Figure 7 a) shows the result after $m\lceil \log m \rceil = 6560$ iterations, clustering cost is 187. Figure 7 b) shows the result after $0.1m = 164$ iterations; clustering cost is 194.6 which is 4% more then a). Table 2 summarizes experiments with the Lucy model. The facility cost was set equal to the bounding box diagonal. You can compare time required for $m \log m$ and $0.1m$ iterations. If we use the reduced number of iterations, the clustering cost is slightly higher in each block. The table records statistics about this error so you can review the impact on clustering quality.



a) b)

Figure 7: Clustering results a) after 6560 iterations, b) after 164 iterations.

8 Conclusion and future work

We have presented a modified data stream approach to solve the facility location problem on geometric data. We suggested an improvement to the facility location algorithm to limit the number of points inspected when computing the gain function. We proposed the facility weight normalization so that clustering works correctly at higher levels. We also performed experiments on clustering geometric data, described algorithm behaviour in various situations and discussed proper settings of particular parameters.

As a future work we would like to use this clustering method to create multiresolution geometric models where the user can select different levels of detail in various parts. It might be also interesting to add topological constraints so that points from different parts of model will not be joined into one cluster. Another interesting task is to cluster a triangular mesh so that clusters are consistent with the mesh topology. We could also take into account distribution of points in the data stream. Points arriving short one after another will be assigned to the same cluster, while points from different parts of the stream (even geometrically close together) will go into different clusters.

References

CHARIKAR, M., AND GUHA, S. 1999. Improved combinatorial algorithms for the facility location and k-median problems. In *IEEE Symposium on Foundations of Computer Science*, 378–388.

CHARIKAR, M., KHULLER, S., MOUNT, D. M., AND NARASIMHAN, G. 2001. Algorithms for facility location problems with outliers. In *Symposium on Discrete Algorithms*, 642–651.

CHARIKAR, M., GUHA, S., ÉVA TARDOS, AND SHMOYS, D. B. 2002. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer System Sciences 65*, 1, 129–149.

CHARIKAR, M., O'CALLAGHAN, L., AND PANIGRAHY, R. 2003. Better streaming algorithms for clustering problems. In *Proc. of 35th ACM Symposium on Theory of Computing (STOC)*, 30–39.

CHUDAK, F. A., AND SHMOYS, D. B. 2004. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Comp. 33*, 1, 1–25.

FRAHLING, G., AND SOHLER, C. 2005. Coresets in dynamic geometric data streams. In *Proceedings of the 37th annual ACM symposium on Theory of computing (STOC)*, 209–217.

GUHA, S., AND KHULLER, S. 1998. Greedy strikes back: Improved facility location algorithms. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 649–657.

GUHA, S., RASTOGI, R., AND SHIM, K. 1998. CURE: An efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 73–84.

GUHA, S., MISHRA, N., MOTWANI, R., AND O'CALLAGHAN, L. 2000. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, 359–366.

GUHA, S., MEYERSON, A., MISHRA, N., MOTWANI, R., AND O'CALLAGHAN, L. 2003. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering 15*, 3, 515–528.

ISENBURG, M., AND GUMHOLD, S. 2003. Out-of-core compression for gigantic polygon meshes. In *SIGGRAPH'03 Conference Proceedings*, 935–942.

ISENBURG, M., AND LINDSTROM, P. 2005. Streaming meshes. In *Proceedings of Visualization'05*, 231–238.

ISENBURG, M., LINDSTROM, P., AND SNOEYINK, J. 2005. Streaming compression of triangle meshes. In *Proceedings of the 3rd Eurographics symposium on Geometry processing (SGP)*, 111.

ISENBURG, M., LINDSTROM, P., GUMHOLD, S., AND SHEWCHUK, J. 2006. Streaming compression of tetrahedral volume meshes. In *Graphics Interface*, 115–121.

ISENBURG, M., LIU, Y., SHEWCHUK, J., AND SNOEYINK, J. 2006. Streaming computation of delaunay triangulations. *ACM Trans. Graph. 25*, 3, 1049–1056.

JAIN, K., AND VAZIRANI, V. V. 1999. Primal-dual approximation algorithms for metric facility location and k-median problems. In *IEEE Symposium on Foundations of Computer Science*, 2–13.

JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. 1999. Data clustering: A review. *ACM Computing Surveys 31*, 3, 264–323.

KAUFMAN, L., AND ROUSSEEUW, P. J. 1990. *Finding groups in data: An introduction to cluster analysis*. John Wiley, New York.

KORUPOLU, M. R., PLAXTON, C. G., AND RAJARAMAN, R. 1998. Analysis of a local search heuristic for facility location problems. In *9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1–10.

LINDSTROM, P. 2000. Out-of-core simplification of large polygonal models. In *Siggraph 20000, Computer Graphics Proceedings*, 259–262.

MAHDIAN, M., MARKAKIS, E., SABERI, A., AND VAZIRANI, V. V. 2001. A greedy facility location algorithm analyzed using

Table 2: Experiments on the number of iterations.

block size [points]	time for $m \log m$ [h:m:s]	time for $0.1m$ [h:m:s]	percent of time used for $0.1m$	min. error	avg. error	max. error
1250	1:01:24	0:02:41	4.4%	0	1.5%	14.8%
2500	1:58:00	0:04:29	3.8%	0.2%	1.6%	5.0%
5000	3:51:01	0:07:45	3.4%	0.5%	1.7%	4.9%
7500	5:43:21	0:11:09	3.3%	0.5%	1.7%	3.5%
10000	7:36:36	0:14:32	3.2%	0.8%	1.6%	3.2%

dual fitting. In *4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 127–137.

MEYERSON, A. 2001. Online facility location. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science (FOCS)*, 426.

MUTHUKRISHNAN, S. 2003. Data streams: Algorithms and applications. In *Proceedings of the 14th annual ACM-SIAM symposium on discrete algorithms*.

NG, R. T., AND HAN, J. 1994. Efficient and effective clustering methods for spatial data mining. In *20th Intl. Conference on Very Large Data Bases*, 144–155.

O'CALLAGHAN, L., MISHRA, N., MEYERSON, A., GUHA, S., AND MOTWANI, R. 2002. Streaming-data algorithms for high-quality clustering. In *18th International Conference on Data Engineering (ICDE)*, 685.

PAJAROLA, R. 2005. Stream-processing points. In *Proceedings IEEE Visualization*, 239–246.

ROSSIGNAC, J. R., AND BORREL, P. 1993. Multi-resolution 3D approximations for rendering complex scenes. In *Geometric Modeling in Comp. Graphics*, 455–465.

SHARIFZADEH, M., AND SHAHABI, C. 2004. Approximate voronoi cell computation on geometric data streams. Tech. Rep. 04-835, University of Southern California, Computer Science Department.

SHMOYS, D. B. 2000. Approximation algorithms for facility location problems. In *Proceedings of International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, 27–33.

STANFORD, 2007. Stanford 3D scanning repository. http://graphics.stanford.edu/data/3Dscanrep/.

ZHANG, T., RAMAKRISHNAN, R., AND LIVNY, M. 1996. BIRCH: An efficient data clustering method for very large databases. In *ACM SIGMOD International Conference on Management of Data*, 103–114.

Improving Introductory Programming Courses by Using a Simple Accelerated Graphics Library

Thomas Larsson and Daniel Flemström
Department of Computer Science and Electronics
Mälardalen University
Sweden

Abstract

We present a platform independent and hardware accelerated graphics library, which has been found to be a suitable educational tool for novice programmers. The purpose of the library is to change the nature of the assignments and projects used in introductory programming courses to something that fascinate and stimulate the students, e.g., game creation. We describe our experiences from using the presented graphics library in four different course instances of our introductory C++ course. The course surveys show that most students found the approach interesting and fun. As faculty, we could clearly see how many students became highly engaged in their projects and some of them accomplished solutions way beyond our expectations. In particular, compared to the programming courses we have given in the past, in which a standard framework for creating window applications was used, we have noticed a significant improvement in terms of the quality of the students' project solutions.

CR Categories: K.3.2 [Computers and Education]: Computer and Information Science Education—Computer Science Education

Keywords: programming, teaching, motivation, graphics, games

1 Introduction

The motivation that drives the students is of vital importance for what they accomplish during their education. This is especially true in programming courses, where students have to spend a lot of time practicing, i.e. writing and debugging code [Jenkins 2001]. But what makes students in introductory programming courses interested and enthusiastic? Despite that there are no simple answers to this question, we have noticed that there are some types of projects that can make a difference. Graphics, multimedia, and game applications seem to fascinate a broad category of students [Guzdial and Soloway 2002]. In particular, many of our students seem to enjoy programming video games.

Therefore, we have experimented with this idea of using a graphics library in four different instances of an introductory course on C++ programming. Students entering the course were assumed to have previous programming experience equivalent to five weeks of training in imperative programming. As a final project in the course, we let the student implement a sprite based video game in 2D. We wanted the complexity of theses games to resemble the complexity of classical games like e.g. Pacman, Asteroids, Space Invaders, or Tetris, including graphics, interactivity, and sound. To make this a reachable goal, we implemented the Simple Accelerated Graphics Library (SAGLib). The main contribution of the library lies in the unique simplicity in the way it provides access to hardware accelerated computer graphics for novice C or C++ programmers. Other existing tools are either designed to be used with other programming languages, have unnecessarily complex APIs for novice students, do not support the creation of fully interactive graphics applications, or do not exploit hardware accelerated graphics.

The results of using our library were mainly satisfactory. In course surveys, as well as in class-room conversations, the students expressed that by using the simple graphics library, the programming experience became visually rewarding, which served as an extra motivation factor for them to produce better and more interesting project solutions. It also turned out that the library in itself introduced no, or very little, extra complexity, compared to programming text-based applications. Thus, the students were able to focus on the design and programming of the application, without worrying about low-level graphics details.

The remainder of this paper is organized as follows. In the next section, relevant prior work is reviewed. Section 3 briefly presents the design of the library and some minimal example applications are given. How the library has been integrated in our introductory C++ course is described in Section 4, and some examples of student projects are also given. Then, in Section 5, the evaluation results as well as the experiences gathered from using the library in our courses are discussed. Finally, our conclusions are given in Section 6.

2 Related Work

Increasing the motivation by using graphics in introductory programming courses is not a new idea. Roberts reports on using a C-based Graphics Library for the first programming course with good results [Roberts 1995]. Another suggested approach supports the creation of simple graphics applications, while keeping the simplicity of text-based programming by letting the compiler automatically create a graphical user interface for programs that otherwise would appear to be text-based. For example, this makes it possible for students to create simple games like tic-tac-toe, checkers, or battleship. More sophisticated graphics programs, however, including e.g. animations, are not possible in this case [Carlisle 1999].

A more powerful approach is to provide a simple graphics library that hides as much of the code complexity as possible that arise when using advanced graphics and windowing systems, but still enables the possibility to create powerful graphics programs. Astrachan and Rodger report that the key benefit of using their library is that interactive graphics generate student interest and enthusiasm, which often lead to increased learning and mastery of the course material. In addition, in many cases, the visual feedback from the programming assignments immediately reveals the existence of bugs, which provides guidance in the debugging process [Astrachan and Rodger 1998].

Childers et al. present EzWindows, which is a simple and portable graphics library for teaching object-oriented programming using C++. The main purposes of their library are to help students grasp the object-oriented paradigm as well as make it possible for students to create programs that resembles the look of other programs in modern desktop computers [Childers et al. 1998].

Rasala argues for the use of toolkits in general in any modern first year computer science curriculum. For example, graphics toolkits are needed because graphics is not a built-in feature in many

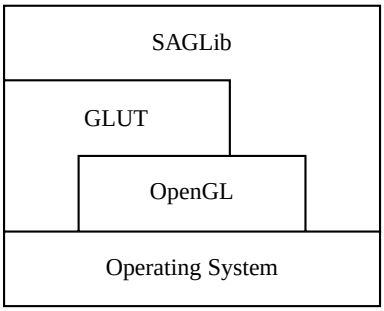


Figure 1: *SAGlib provides a complete layer that hides the operating system, GLUT, and OpenGL for the application programmer.*

languages. Graphics is necessary for many important computing activities, helps in the debugging process by revealing errors, and students are attracted to the discipline by computer-generated images [Rasala 2000].

The integration of a multimedia project in basic programming courses that emphasizes alternative learning styles through collaborative team work has also been proposed [Wolz et al. 1997]. In the media computation approach, suggested by Guzdial as a response to the widespread use of traditional, overly-technical, and abstract course content, all programs written manipulates sound, images, and movies to increase student motivation and hopefully enabling a deeper learning [Guzdial 2003; Guzdial and Soloway 2002]. The leading stars behind the approach were relevance (concrete domain examples), creativity (open-ended assignments), and social (collaboration and sharing of experiences) [Tew et al. 2005].

Meyer presents a new course design for novice programmers using the language Eiffel together with a graphics library called TRAFFIC, which includes some basic elements of a Geographical Information System [Meyer 2003]. The immediate usage of well-designed interfaces, APIs, hiding simple to use, but internally sophisticated, functionality is captured by the name Outside-In, or the Inverted Curriculum, rather than traditional bottom-up or top-down approaches. Data gathered from two course instances indicate student appreciation and higher grades compared to previous course versions [Pedroni and Meyer 2006]

Furthermore, we note that graphics and games are not the only way of letting students work on for them more meaningful and concrete projects. Another good example is to use programming of robots to concretize the projects and stimulate student interest [Lawhead et al. 2003].

Of course, there are many other factors besides the usage of appropriate toolkits and libraries that instructors need to take into account to be successful in teaching introductory programming, for example the choice of programming languages, programming environments, classroom techniques, and type of examination. These factors, however, will not be discussed in this article. An interesting taxonomy of the first two factors, however, has been given by Kelleher and Pausch [2005].

3 Library Design

Roberts [1995] proposed the following four important interface design criteria for their graphics library:

i It must be simple.

ii It must correspond to student intuition.

iii It must be powerful enough for students to write programs they think are fun.

iv It must be widely implementable.

Furthermore, we can add that the usage of a library in a course should be motivated by it giving a direct or indirect benefit with respect to the course objectives.

When designing SAGLib, these were the design criteria we had in mind. Therefore, we decided to provide only a limited number of simple functions. The API hides the inner complexity of OpenGL and GLUT. This makes it possible for the students to focus on data structures and programming rather than graphics programming. This is illustrated in Figure 1. Note that the application programmer only needs to know anything about SAGlib to create simple accelerated graphics applications. Both the C and C++ API provide functionality for

- setting up and displaying a window,
- controlling pens, colors and transparency,
- drawing bitmaps and shapes, with or without simple geometry transformation, and
- simple event handling for mouse, keyboard, resize, and redisplay

For simplicity, our graphics interface is mainly restricted to 2D graphics applications. It supports the drawing of basic 2D shapes, bitmaps, and text. An event-based execution model is used, together with basic timer functionality to support animation. The library also provides basic mouse and keyboard input handling. In our simple setting, this functionality makes the creation of simple video games, or the creation of other types of media rich applications, a reachable goal, even for novice programmers. We note, however, that it has also been demonstrated, through the ALICE system, that novice programmers can develop interactive 3D graphics and animation as part of their initial learning of programming concepts [Cooper et al. 2000].

To make the library useful for courses using the imperative as well the object-oriented programming paradigm, we use a two layer API, one in C and one in C++. Also, since our library is implemented on top of standard OpenGL, it becomes very powerful considering the huge improvements over the last few years in commodity graphics hardware. This also means that the library can make use of special 3D graphics functionality internally, for example texture mapping and alpha blending, as appropriate. This approach also make the library portable. Furthermore, we use GLUT for basic window handling [Kilgard 1996], which also is supported on many platforms. Both OpenGL and GLUT, however, are invisible for users of our library[1]. The latest version of our library is freely available on the web [Larsson and Flemström 2006]

Hardware acceleration is automatically enabled since SAGLib is based on OpenGL. We use an orthographic projection with a one-to-one mapping between world positions and pixel coordinates. No z-values need to be specified; they are assumed to be zero. Thus, the 2D graphics image will be specified in the x-y plane.

OpenGL was chosen for several reasons. First of all, we wanted access to the high performance available in commodity graphics hardware in a simple way. Many of the accelerated functions supported in 3D graphics hardware is very useful also in 2D graphics applications, for example double buffering, geometry transformation, primitive rasterization, texture mapping and alpha blending.

[1] It is still possible for skilled students to access OpenGL functionality directly if so wanted.

Function names	
sgDrawPoint	*sgDrawLine*
sgDrawRect	*sgDrawFilledRect*
sgDrawCircle	*sgDrawFilledCircle*
sgDrawTriangle	*sgDrawFilledTriangle*
sgDrawQuad	*sgDrawFilledQuad*
sgDrawText	

Figure 2: *The drawing functions supported in SAGLib.*

The tremendous performance of today's graphics accelerators, also means that everything can be redrawn at each display update without much performance penalty. Furthermore, the graphics rendering is double buffered for flicker-free animation. All this simplifies for the student who can concentrate on the main programming tasks instead of, for example, optimizing the graphics drawing strategy.

The geometrical drawing functions that are supported are listed in Figure 2. These basic shapes are drawn with the current pen and color. An alpha value is also associated with each chosen color. It can be specified as a value between $0 - 255$; that is, from fully opaque to fully transparent. The pen, transparency and color selections will be used until changed by new API calls. This reduces the number of arguments to the functions and the complexity of the API.

Since each color drawn may be transparent, and alpha blending is always enabled, many exiting effects can be created with little effort, for example, sprite animation and clouds. Transparent bitmaps are often unnecessary complex to achieve in other programming environments. In SAGLib, we have simplified internal representation of graphic formats by only using the true color format with 32 bits (RGBA) per pixel. Each pixel may have its own transparency. Thus, a bitmap may have arbitrary many transparent colors.

Also, since font handling usually is quite complex, text output has been reduced to one fixed font. Some students, however, have created their own text styles by drawing custom designed bitmaps. To reduce complexity even further, only one window is allowed per application and we have pre-defined the major application settings, just leaving the window size, position and caption to the user. All event handling has been hidden within the library. The user creates a display drawing function, which the library use as a callback routine. In this function, the API functions for pens, colors, transparency, bitmaps, and shapes may be used. Also, the system calls the display function automatically when for example the user moves or resizes the window.

Since threading is a very complex issue, we have reduced the number of timers to one, which means that the application will run in one thread. In the callback routine of the timer, the code to calculate and update the positions of e.g. moving shapes or sprites should be placed. Together with the support for drawing transparent bitmaps, the timer functionality makes it very simple for the students to include sprite animations in their applications.

3.1 Creating Applications

To make the library more widely usable, it includes support for both C and C++ programming. The C API encapsulates and hides the operating system, GLUT and OpenGL from the application programmer. An event-based execution model is used, which means that the application programmer can subscribe on events by registering callback functions. There are support for keyboard, mouse, redraw, resize, and timer events. An example of a minimal graphics application in C that draws a rectangle in the application's window

```
void myDisplay(void) {
  sgClearDisplay();
  sgDrawFilledRect(500, 20, 200, 230);
  sgFinishDisplay();
}

void main() {
  sgInitGraphics(400, 400,"MyApp", myDisplay);
  sgMainLoop();
}
```

Figure 3: *A minimal C program using SAGLib.*

```
class MyApp:public Application {
public:
  int cx;
  Bitmap bitmap;

  MyApp(void) : cx(0) {
    bitmap.initFromBMP("player.bmp");
    bitmap.addTransparentColor(255,255,255);
  }

  void onDraw(Graphics& g) {
    g.setColor(200, 50, 0);
    g.drawFilledRect(50, 50, 50, 50);
    bitmap.draw(g, cx, 100);
  }

  void onKey(unsigned char key,int x,int y) {
    if(key == 'l') cx = cx + 3;
    if(key == 'k') cx = cx - 3;
    updateDisplay();
  }
};

void main(void) {
  MyApp app;
  app.init("MyApp");
  app.showModal();
}
```

Figure 4: *A simple C++ program using SAGLib.*

is shown in Figure 3.

The C++ level completely encapsulates the C-level API which means that the students can use object-oriented programming throughout their entire applications. The C++ API consists of three classes. *Application*, *Graphics* and *Bitmap*. Each user application inherits from the Application class. The Application class sets up the window and handles the window events, such as mouse, keyboard, drawing events, timers and so on. Subscriptions for available events are automatically set up in the Application class by the usage of virtual member functions. You may also check the state of a specific key with the function *Application::isKeyDown* which allows games where several keys might be pressed at the same time.

The *OnTimer* method in the *Application* class should be overridden to handle position updates, calculations etcetera. Last in this method, a call to *Application::updateDisplay* will raise the *OnDraw* event. The Graphics class handles double buffering automatically and gives access to the accelerated graphics primitives in an object oriented way. All primitive graphics functions from the C-level API can be found as class methods on the Graphics object.

The example application in Figure 4 creates a window on the

screen, loads a bitmap from file and allows the user to control it with the keyboard. The bitmap acts as a simple sprite, where the color white is specified as completely transparent. It can be noted that the rectangle drawn before the user-controlled bitmap appear to be behind it at all times.

4 Course Design

Here we will present some details of how we have used SAGLib in our C++ course. The purpose of the course is to teach the fundamentals of the C++ language. In the course syllabus the following topics are included:

- Control structures: selection, iteration, recursion
- Functions
- Arrays, pointers, and strings
- Classes and objects
- Operator overloading
- Inheritance
- Dynamic binding
- Templates
- Data structures and algorithms in the Standard Template Library.
- IO streams and file processing

The course objectives state that students, upon completion of the course, will be able to

- Understand the differences between imperative vs. object-oriented programming.
- Writing object-oriented programs in C++ on their own that demonstrate mastery of object-oriented concepts such as abstract data types, encapsulation and information hiding, function and operator overloading, inheritance, and polymorphism.
- Understand the execution model of event-based programs with graphical user interfaces.

Some examples of topics that we usually only touch upon briefly in the course are exception handling, name spaces, and object-oriented analysis and design. Note, however, that these are not programming skills that were removed from the course because of the inclusion of our graphics library. The only thing that we have removed from earlier incarnations of the course is the usage of another more advanced class library for the creation of graphical user interfaces.

The course included lectures, laboratory work, take-home exercises, a project task, and a final written exam. The graphics library was introduced right from the start of the course. An overview of the library and some initial examples were presented already on the first lecture. Also, some initial exercises based on SAGLib were carried out in the first laboratory session.

For example, a concrete and visually rewarding way to practice iteration together with two dimensional arrays is simple image or bitmap manipulation. Some examples of operations that are suitable to be implemented by novice programmers are image color inversion, vertical and horizontal flip, and color to gray scale conversion. An example of the first operation is given in Figure 5. In this case, the inverted RGB color, I, for each pixel is simply given as

$$\begin{aligned} I_{red} &= 255 - S_{red} \\ I_{green} &= 255 - S_{green} \\ I_{blue} &= 255 - S_{blue} \end{aligned}$$

where S is the RGB color of the corresponding pixel in the source image. Other slightly more advanced exercises would be to implement linear filtering such as blur, sharpen and edge detection. These operations involves calculating a new pixel color as a weighted average of a neighborhood of pixels in the source image.

Figure 5: *Results of a simple image color inversion operation.*

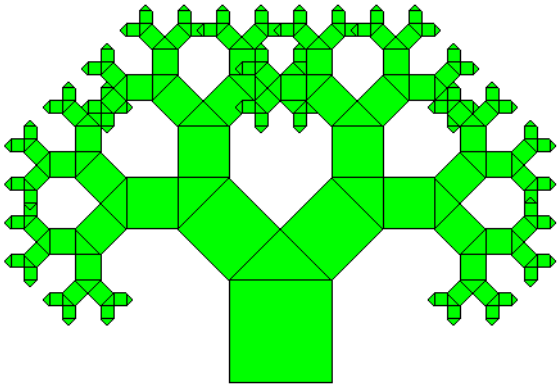


Figure 6: *The Pythagoras Tree — a simple example of a recursively generated image.*

When teaching recursion, which is a concept that many computer science students perceive as difficult, naturally visual examples may contribute to the students' learning and appreciation of this concept. Some concrete examples are various forms of simple fractal images such as the Pythagoras tree, the Koch snowflake, and the Sierpinski triangle (or gasket). In Figure 6, an image of the Pythagoras tree is shown which has been produced by a simple SAGLib program that we use as a code example when teaching recursion. Another interesting recursion example that we have used is a simple version of the flood fill algorithm for painting the interior of arbitrary shapes in a pixel-based image.

Year 2003									
Q	N	R	1	2	3	4	5	Average	Median
Q1	29	18	1	0	1	12	4	4.00	4
Q2	29	15	0	0	6	3	6	4.00	4

Year 2004									
Q	N	R	1	2	3	4	5	Average	Median
Q1	21	12	0	0	3	1	8	4.42	5
Q2	21	12	0	1	6	0	3	3.50	3

Year 2005									
Q	N	R	1	2	3	4	5	Average	Median
Q1	51	31	0	0	6	14	11	4.16	4
Q2	51	26	1	2	11	5	7	3.58	3

Year 2006									
Q	N	R	1	2	3	4	5	Average	Median
Q1	22	14	0	1	2	7	4	4.00	4
Q2	22	14	1	1	4	3	5	3.71	4

Table 1: *Results from course surveys during 2003-2006. Note that N is the number of enrolled students and R is the number of responses we received.*

Later on, somewhere in the middle of the course, take-home assignments are given out in which the students create a very simple example game step by step in order to learn to understand timer events, drawing simple shapes and keyboard handling. In another preparatory assignment, the state pattern is used to implement a rudimentary animated submarine game. The submarine has different animations depending on how it moves on the screen. The purpose of this is to learn handling interactive graphics and game character states.

4.1 Student Projects

As the major programming task or project, the students are given the opportunity to create a simple action game that includes sprite animation, several game levels, saving and loading options, and optionally a high score list and sound effects. The students are also allowed to design a game idea more freely of their own, if they so wish. In this way, they get the opportunity to choose a task that really interest them, potentially increasing their motivation and creativity further. They are required, however, to get their design approved by the course leader to make sure that the level of difficulty of the implementation seems reasonable with respect to the goals of the course. Screen shots from some student projects are shown in Figure 7. Other examples of projects the students have accomplished are remakes of early home computer games such as Manic Miner and Jet-Pack, helicopter and aeroplane games, and sports games.

5 Evaluation and Discussion

The proposed approach has been employed in four course instances during 2003–2006. Evaluation was carried out by course surveys at the end of each course. The following survey questions were used:

Q1 How would you rate the overall quality of this course (1-5)?

Q2 How would you rate the usage of SAGLib in the course (1-5)?

Q3 In what ways did you benefit from the project task?

Q4 What were the strongest features of this course?

Q5 What were the weakest features of this course?

Q6 How would you like to see this course changed in the future?

There were also some additional questions on the survey where the students could give comments on the lectures, assignments, and exercises, and also, they were encouraged to give advices to the teachers. The survey results from the first two questions are given in Table 1.

Regarding the usage of SAGLib, some characteristic positive comments given as answers to the other questions (Q3–Q6) are:

- SAGLib was fun. Even something as simple as incrementing a variable becomes fun if something is happening on the screen.
- SAGLib has been working very well. Good structure and childishly simple to use.
- It's fun with graphics because of the visible results one gets.
- SAGLib is user friendly.
- I think SAGLib is the best I ever have seen.
- It was nice to get graphical results from one's hard work, instead of the usual command prompt.
- SAGLib made it easier so that one could focus on the important things.
- It was a pleasure to work with a rather simple graphical user interface.
- The project was good, making games instead of making programs.

Many of these comments were repeated with some variations by other students. From all the given comments, it was clear that a strong majority of the students expressed that the usage of SAGLib benefitted the course and their own learning experience in one way or another. To exemplify how fond of the library some of the students have become, we can mention that several students have come visiting us after the course asking if they are allowed to use the library for making their own games at home.

Over these years, only a handful of students have expressed some negative things or feelings about SAGLib. As the following comments show, some of the more skilled students wanted to use more advanced graphics APIs with more features. Unfortunately, some students also ran into problems in their project because of some bugs that appeared in the graphics library, which we could feel made them dislike the library. Here are some characteristic criticisms which were expressed in the surveys:

- SAGLib doesn't feel perfect. Some features are missing.
- I don't understand why we must use SAGLib when there are better well-documented alternatives.
- The graphics becomes slow if one uses many and relatively large images with SAGLib.
- SAGLib was buggy.

The course evaluations, together with our own experiences from interacting with the students during these courses, have made it evident for us that the usage of SAGLib has improved our introductory C++ course. In particular, we could see that the combination of take-home assignments and usage of SAGLib that we used was highly appreciated. It was a strong correlation between the students that finished their take-home assignments and the students that finished their projects on time. Also, at the end of the courses, a final written exam was given, which showed that most students

Figure 7: *Screen shots from four different student projects.*

were able to use their acquired skills, from the take-home assignments and their projects, to solve relevant programming problems satisfactorily.

Over the years we have tried to use OWL (Borland's Graphics Library), MFC and straight WIN32 programming for the final project in our introductory C++ course. All of these libraries required that the student more or less got a finished application to modify. This was perceived to be rather difficult and there was a lot to learn about all classes in respective library. Much time was consumed by trying to use the wrong class in the wrong context. For example, to use transparent bitmaps, students were forced to write or cut and paste code they did not fully understand. The supposed to be fun event driven graphics programming and C++ design were obscured by these frameworks. With SAGLib, on the other hand, there are few functions and classes to master, which means that the students were able to concentrate on solving the intended problem. Since they could concentrate on design and C++ issues, we could see how the quality of their solutions was increased significantly compared to previous years.

We would also like to point out that today's students are well aware of the advanced graphical user interfaces that most applications have in modern windowing systems. If they want to learn programming, but are only allowed to create programs using simple textual input and output they might get discouraged. Furthermore, the usage of graphics in programming might help the instructor in creating a feeling among the students that programming is fun. This can raise the students interest and enthusiasm and make them work harder and learn more. Another benefit of using a graphics system in learning programming is the direct visual feedback you can get as a result of your programming, which may be very helpful when debugging. Another possibility is to use graphics to visualize how algorithms work. Special code for algorithm animation can be provided by the instructor or the students themselves can produce an algorithm animation as a programming exercise.

A possible drawback of using a graphics library in introductory programming courses would be that learning to use the library could consume too much time, and so forcing the instructor to remove other important concepts or exercises from the course. It is therefore important that the graphics system is simple to use and that it provides a fruitful base for learning the programming abilities the course is meant to teach. It is the task of the library to make the programming work easier, and a more joyful experience, not to lay an additional burden upon the students. We have found that SAGLib fulfills these goals extremely well.

Some faculty may also argue that it is better to focus on using functions and classes that are supported in international standards than on using a non-standardized graphics API in introductory courses. Many programming principles and solutions, however, may very well be illustrated by considering the design of the graphics library itself. Also, programmers in industry use different frameworks, toolkits and libraries on a daily basis [Rasala 2000].

Regarding survey question Q6 about future changes for the course, a student wrote: "Don't force everyone to make games". It is true that the library mainly has been used for simple game programming in our courses, but nothing really hinders the library to be used outside this context to be more appealing for a broader category of students. Therefore, we are planning to emphasize the open-ended nature of the programming project more in the future to make it clear that students really have the opportunity to be creative also outside the game programming context.

The good project results we have observed can partly be explained by the fact that the presented course was not the students very first programming course. They were novice C++ programmers, but they had taken a first course in an imperative programming language previously. Therefore, it would also be interesting to try using SAGlib in a course where the students have no earlier programming experience at all. Maybe a somewhat simpler project would be required in such a case.

Interestingly, we have been told that SAGLib already has been used in some programming projects in upper secondary school in cooperation with our university. Also in this case, simple games with 2D graphics, e.g. Tetris, were implemented successfully, and the pupils were reported to be satisfied with the library [Nolte 2006].

Before the pupils entered the project, however, they have already been briefly introduced to rudimentary programming of text-based applications at their school.

6 Summary and Conclusions

It is true that there are many difficulties students encounter while learning basic programming skills [Jenkins 2002]. We believe, however, that if students find the programming assignments inspiring, it can motivate them to overcome many of the arising problems.

To be successful, instructors need to consider the motivating factors that drive the students. In fact, many students in computer science find computer graphics to be one of the most interesting subjects in the curriculum [Carlisle 1999]. Also, applications in modern operating systems are expected to have sophisticated graphical user interfaces. Thus, it makes sense to allow the students to learn programming by using a simple graphics API.

When we used our simple accelerated graphics library in our introductory programming courses, we experienced very encouraging results. By providing a simple way to create graphics applications many students show more interest and enthusiasm in the programming exercises. In particular, we have found that the possibility to create arcade-style video games, with fast, flicker-free sprite animation, through hardware accelerated graphics attracts many students.

Of course, incorporating a graphics library into basic programming courses must be done in a careful and proper way to ensure that usage of graphics enrich the learning experience. It is important to note that the usage of the library must not introduce extra code complexity that risks to take the students focus away from the basic programming skills they try to learn. Because of the simplicity of our library, however, we have not noticed any such negative effects.

References

ASTRACHAN, O., AND RODGER, S. H. 1998. Animation, visualization, and interaction in CS 1 assignments. In *SIGCSE '98: Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education*, ACM Press, New York, NY, USA, 317–321.

CARLISLE, M. C. 1999. Graphics for free. *ACM SIGCSE Bulletin 31*, 2, 65–68.

CHILDERS, B., COHOON, J., DAVIDSON, J., AND VALLE, P., 1998. The design of EzWindows: A graphics API for an introductory programming course.

COOPER, S., DANN, W., AND PAUSCH, R. 2000. Alice: a 3-D tool for introductory programming concepts. In *CCSC '00: Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges*, Consortium for Computing Sciences in Colleges, USA, 107–116.

GUZDIAL, M., AND SOLOWAY, E. 2002. Teaching the Nintendo generation to program. *Commun. ACM 45*, 4, 17–21.

GUZDIAL, M. 2003. A media computation course for non-majors. In *ITiCSE '03: Proceedings of the 8th annual conference on Innovation and technology in computer science education*, ACM Press, New York, NY, USA, 104–108.

JENKINS, T. 2001. The motivation of students of programming. In *ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education*, ACM Press, New York, NY, USA, 53–56.

JENKINS, T. 2002. On the difficulty of learning to program. In *Proceedings of the 3rd annual LTSN-ICS Conference*, 53–58.

KELLEHER, C., AND PAUSCH, R. 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv. 37*, 2, 83–137.

KILGARD, M. J., 1996. The OpenGL Utility Toolkit (GLUT) programming interface, API version 3, November.

LARSSON, T., AND FLEMSTRÖM, D., 2006. SAGLib: Simple Accelerated Graphics Library. http://www.idt.mdh.se/SAGLib.

LAWHEAD, P. B., DUNCAN, M. E., BLAND, C. G., GOLDWEBER, M., SCHEP, M., BARNES, D. J., AND HOLLINGSWORTH, R. G. 2003. A road map for teaching introductory programming using LEGO mindstorms robots. *SIGCSE Bull. 35*, 2, 191–201.

MEYER, B. 2003. The outside-in method of teaching introductory programming. In *Ershov Memorial Conference, volume 2890 of Lecture Notes in Computer Science*, M. Broy and A. V. Zamulin, Eds. Springer-Verlag, 66–78.

NOLTE, T., 2006. Personal communication.

PEDRONI, M., AND MEYER, B. 2006. The inverted curriculum in practice. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, ACM Press, 481–485.

RASALA, R. 2000. Toolkits in first year computer science: a pedagogical imperative. In *SIGCSE '00: Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*, ACM Press, New York, NY, USA, 185–191.

ROBERTS, E. S. 1995. A C-based graphics library for CS1. In *SIGCSE '95: Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education*, ACM Press, New York, NY, USA, 163–167.

TEW, A. E., FOWLER, C., AND GUZDIAL, M. 2005. Tracking an innovation in introductory CS education from a research university to a two-year college. In *SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*, ACM Press, New York, NY, USA, 416–420.

WOLZ, U., DOMEN, D., AND MCAULIFFE, M. 1997. Multimedia integrated into CS 2: an interactive children's story as a unifying class project. In *ITiCSE '97: Proceedings of the 2nd conference on Integrating technology into computer science education*, ACM Press, New York, NY, USA, 103–110.

Visualisation of human characteristics in vehicle and health care product development

Mikael Blomé
Lund University
mikael.blome@design.lth.se

Lars Hanson
Lund University
lars.hanson@design.lth.se

Dan Högberg
University of Skövde
dan.hogberg@his.se

Maria Jönsson
Arjo R&D Center
maria.jonsson@arjo.se

Daniel Lundström
CARAN AB
dalus@wmdata.se

Dan Lämkull
Volvo Car Corporation
dlamkull@volvocars.com

Abstract

The purpose of the research project described in this paper is to improve the efficiency of product development processes by exchanging knowledge and experiences about user centred design methods and technologies between the two branches: vehicle and health care industries. The health care industry can benefit from visualisation and simulation tools that include computer manikins, a physical representation of the human, and the vehicle industry can benefit from manikins having personal characteristics, which has proven to be successful in the health care industry.

Keywords: visualisation, simulation, human characteristics, user representation, product development

1 Introduction

Visualisation in industry is often referred to as the process and result of illustrating information. This can be done with different techniques, such as CAE (Computer Aided Engineering) and simulation software. A key advantage of visualisation is that it enables an illustrative presentation of phenomena, such as the overview and relationships of processes and products. It has also been established that visualisation can work as a common language between persons with different backgrounds and specialities, thereby supporting communication and exchange of knowledge in a multidisciplinary and collaborative working manner [Blomé, 2004].

The vehicle industry can be seen as a representative example in this context, where multidisciplinary organisations develop complex products with a clear human-product interface, and where there is a need among specialists to agree on an appropriate balance of a range of product requirements for the final vehicle design and its production system. Not surprisingly, the vehicle industry is a pioneer when it comes to using visualisation techniques. Their vision is to develop and evaluate the product and production systems in virtual environments more or less throughout the entire development process [Lämkull, 2006]. Visualisation and simulation techniques are employed when possible to evaluate conceptual and detailed solutions without the need for physical prototypes. This makes it possible to increase the number of evaluations in the design process compared with the traditional approach based on expensive and time consuming physical prototypes and evaluations. The objective is to save time and money, but also to result in better vehicles (products) and more efficient and ergonomic production [Chaffin, 2001]. The virtual approach is considered more efficient and effective, provided that the simulations are performed correctly and based on relevant and correct data [Ziolek and Nebel, 2003].

Technical vehicle performance and manufacturability has traditionally been the focus in the vehicle development process. Most companies acting on the market today, however, meet these basic demands. Customer appreciation of additional product properties, such as aesthetics and attractiveness, has had an increased impact on products' (vehicles'), commercial success [Jordan, 2000]. Furthermore, vehicle assemblers' health and job satisfaction have been highlighted in the production system design process [Falck, 2007]. Since the targeted customer group and assembly personnel often are complex when it comes to personal requirements and expectations, it is hard to estimate and describe the diversity of the groups, and hence to simulate the interaction between the product and humans. This complicates the design process, and the difficulty is significant with respect to the range of personal and emotional characteristics among the users.

The health care branch has increased the visual and virtual approach in product and production development, but this is mostly related to the interaction between the user and the product. It is based on descriptions and visual illustrations of archetypical users, without utilising computer based simulation technology. Still, the users' characteristics are emphasised since the main function of the products is to support everyday life for a wide range of individuals with disabilities or impairments, which often call for different design solutions. The health care industry approach is in line with the vehicle industry's because it too aims to employ computer based visualisation and simulation technology to enhance the product development process efficiency and product quality.

2 Current user representation principle

The present methods for visualising human characteristics in the product development process are described in the following sections, in the health care and vehicle industry respectively. The descriptions are based on interviews with key persons as well as experiences of the research group.

2.1 Health care industry

Arjo, the health care company linked to this project, has been involved in the development of equipment and working techniques in elderly care for over 50 years. Their development activities are based on mobility issues. They have a tradition of working in close cooperation with staff and planners of elderly care facilities, focusing on both the caregiver and resident. Some years ago Arjo developed the *Resident Gallery*, a communication tool based on five levels of functional mobility: from totally mobile and independent residents to those who are entirely bedridden. This system was initially developed to support the product development process and the communication between professionals with different competencies in the development process. Today, the *Resident Gallery* is also used as a sales tool to help customers accommodate their facilities with the right equipment. Figure 1 illustrates the five typical user characters used by the company.

Figure 1: User characters in Arjo product development, classified according to their degree of mobility. (Courtesy of Arjo)

One can identify the most mobile to the most dependent resident by their alphabetical names: Albert, Barbara, Carl, Doris and Emma. A detailed description of characteristics and background is linked to each character, which includes information about age, weight, mobility and individual traits. Using this system and all its benefits led Arjo to the next step, the development of the *Mobility Gallery*. That gallery includes additional characters to accommodate other settings, such as hospitals, special care and home care. These characters are used initially in the design process, but currently have no natural place in the subsequent development process. Therefore, Arjo requested a method in which these characters can be applied in the design, modification, visualisation and analyses of health care equipment. The use of the characters (also known as *personas*) has been successful and the method supports the designers since the characters' situations are easier to visualise and to discuss. Similar effects are reported in Pruitt and Grudin [2003]. The characteristics of the different types of customers have spread from the design department and are now also being used by sales people and buyers. For example, purchasers from health care institutions express their needs of products by describing their patients as four Alberts, six Barbaras, two Carls, one Doris and one Emma.

2.2 Vehicle industry

The Swedish vehicle industry has a well established tradition of using computer manikins – a replica of the human – to design the vehicle-driver interface to suit potential users [Hanson, 2004], but also of adapting production equipment and working tasks to the conditions of the production personnel.

Figure 2: Manikin for ergonomic vehicle interior design. (Courtesy of Saab Automobile)

Figure 2 illustrates a female computer manikin testing the interior of a Saab 9-3. The female manikin can then "tell" the designer how comfortable the driving position is, if it is possible to reach all instruments and what she sees in the mirror, etc. Figure 3 illustrates a male computer manikin testing the assembly of an antenna on a Volvo C30. The male manikin can "tell" the designer about the physical workload on his back and shoulders and if the visual requirements are met for the assembly.

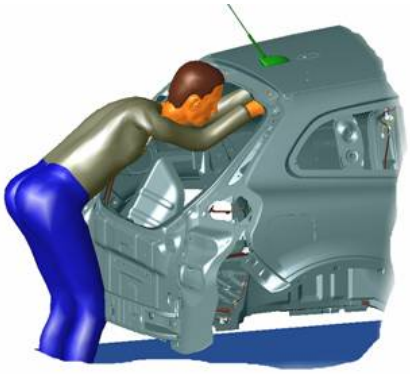


Figure 3: Manikin for evaluating vehicle assembly ergonomics. (Courtesy of Volvo Car Corporation)

The manikins can show this type of information with numbers, but the simulation engineers usually choose to view the results as pictures and colour coded areas. This can, for example, be the view as seen from the eyes of the manikin, or a red coloured shoulder indicating heavy workload, or a coloured area indicating the maximum reach of the manikin.

Visualising information and simulation is a successful approach since it supports communication within and between different disciplines in the vehicle development process. If problems are mutually understood by ergonomists, designers, production engineers and managers, modifications of the product or the workplace are more easily agreed upon. Visualisation is therefore an important approach towards fulfilling drivers' requirements of comfort and control in the vehicle, or healthy and effective assembly on the production line. [Hanson, 2004; Lämkull, 2006]

The manikins utilised to evaluate vehicle interiors or assembly activities aim to represent potential customers of the vehicle or assembly personnel. This is a difficulty for the simulation engineers since each individual is unique in several ways. The current approach used by simulation engineers is to let the manikin represent the bodily (physically) side of the human only. Hence, the manikin is described in anthropometrical terms, typically determined according to corresponding stature. The software defines the remaining body dimensions in order to build up a "normal" person, using existing correlation data. Sometimes a slightly more sophisticated method is utilised where stature, corpulence and proportion are entered to define the manikin's anthropometry. In the literature and at vehicle companies there are strategies for how a reduced number of manikins can represent the anthropometric diversity that exists among vehicle customers and production personnel [Högberg, 2005; Wirsching and Premkumar, 2007]. However, movability is not taken into consideration even though it is possible to alter the movability in the different joints of the manikin. As a result, it is always healthy manikins without any impairments that test the vehicles or the assembly tasks at the virtual (computer simulated) stages of the design process.

The vehicle industry uses descriptions of representatives of the market segments they target in terms of customers' buying power, driving style or hobbies, for example. These characteristics are not reflected in the manikins used and visualised during the development work. However, researcher such as Högberg and Case [2006] as well as Alexander and Conradi [2007] have promoted the introduction of personal characteristics on manikins.

3 Rationale and projected outcome

Based on the descriptions of the situations in the vehicle and health care industries, mutual benefits can be gained from sharing and integrating knowledge and experiences from the two in the consideration and visualisation of user aspects in product development. Introducing efficient visualisation and simulation tools that include computer manikins from the vehicle industry into the health care industry is one such benefit. In the opposite direction, the vehicle industry can benefit from manikins having personal characteristics, which has proven to be successful in the health care industry (but without the use of computer manikins). In essence, the purpose of the research project is to improve the efficiency of product development processes by exchanging knowledge and experiences about user centred design methods and technologies between the two branches: vehicle and health care industries.

3.1 Vehicle industry

After this project the vehicle industry is projected to have a more human centred design process. Although the vehicle industry uses computer manikins in the design process for visualisation, the staff involved frequently treat manikins as anonymous physical objects only. The manikin has the same status as a battery. Both require space in the vehicle to fit either in the cockpit or under the hood. By giving the manikins personal characteristics, based on the experiences from the health care industry, the vehicle development engineer's attitude to them can change. In future discussions, supported by the visualisation of manikins with mapped characteristics, the manikins are more likely to be treated as representing humans. Such a shift may affect the design of the products and product systems positively for the drivers, passengers and assembly personnel. For example, a larger number of users may be accommodated by a more careful and adaptive product or production system design, thanks to the user centred methods and tools employed. Furthermore, there may be a higher degree of customer satisfaction due to the user centred approach in the product design process. Pictures or animations of manikins with personal characteristics using the product or working on the assembly line are expected to encourage and enrich understanding, empathy and communication of user diversity between professionals in different competence groups involved in the development process (e.g. marketing, industrial design, product development, manufacturing engineering and production staff).

3.2 Health care industry

After this project the health care industry will have a formalised evaluation procedure where the results are visualised using computer manikins with personal characteristics. The computer support is expected to result in shorter product development lead time, fewer major iterations and enhanced product quality. The number of physical prototypes will decrease when visualisation and simulation tools are used as a complement to verifying physical prototypes. The personal characters, now sporadically used in early design stages, will be used throughout the entire design process. Such a user centred design process, supported by manikin visualisations can lead to products that to a higher degree meet the demands of the users. Using a set of manikins in the design process for evaluating health care products can lead to products that consider and cater for user diversity in a richer way. A visualisation of a manikin with personal characteristics using health care products is projected to provide enhanced communication between different stakeholders in the health care development process.

Acknowledgements

The project will be carried out at the Virtual Ergonomics Centre (www.vec.se) and financially supported by the Knowledge Foundation (KK-stiftelsen) of Sweden as well as the participating organisations. This support is gratefully acknowledged.

References

ALEXANDER, T. AND CONRADI, J. (2007). *Modeling Personality Traits for Digital Humans*, SAE Technical Paper 2007-01-2507. Warrendale, USA: Society of Automotive Engineers.

BLOMÉ, M. (2004). *Visualization of Guidelines on Computer Networks to Support Processes of Design and Quality Control*. Doctoral thesis. Lund, Sweden: Lund University.

CHAFFIN, D. (2001). Introducion. In: *Digital Human Modeling for Vehicle and Workplace Design*, D. Chaffin (Ed.). Warrendale, USA: Society of Automotive Engineers, pp. 1-14.

FALCK, A-C. (2007). *Virtual and Physical Methods for Efficient Ergonomics Risk Assessments – A development process for application in car manufacturing.* Licentiate thesis. Göteborg, Sweden: Chalmers University of Technology.

HANSON, L. (2004). *Human Vehicle Interaction. Drivers' Body and Visual Behaviour and Tools and Process for Analysis*. Doctoral thesis. Lund, Sweden: Lund University.

HÖGBERG, D. (2005). *Ergonomics Integration and User Diversity in Product Design*. Doctoral thesis. Leicestershire, UK: Department of Mechanical and Manufacturing Engineering, Loughborough University.

HÖGBERG, D. AND CASE, K. (2006). Manikin Characters: User Characters in Human Computer Modelling. In: *Contemporary Ergonomics*, P.D. Bust, (Ed.). UK: Taylor & Francis, pp. 499-503.

JORDAN, P.W. (2000). *Designing Pleasurable Products: An introduction to the new human factors.* London: Taylor & Francis.

LÄMKULL, D. (2006). *Computer Manikins in Evaluation of Manual Assembly Tasks*. Licentiate thesis. Göteborg, Sweden: Chalmers University of Technology.

PRUITT, J. AND GRUDIN, J. (2003). Personas: practice and theory. *Conference on Designing for User Experiences,* San Francisco. New York: ACM Press, pp. 1-15.

WIRSCHING, H-J. AND PREMKUMAR, S. (2007). *Statistical Representations of Human Populations in Ergonomic Design*, SAE Technical Paper 2007-01-2451. Warrendale, USA: Society of Automotive Engineers.

ZIOLEK, S. AND NEBEL, K. (2003). *Human modeling: Controlling misuse and misinterpretation*, SAE Technical Paper 2003-01-2178. Warrendale, USA: Society of Automotive Engineers.

Some remarks about geometry in medicine

Krzysztof T. Tytkowski
Geometry and Engineering Graphics Centre
Silesian University of Technology, Poland
Krzysztof.Tytkowski@polsl.pl

Summary

Geometry has been present in medicine in various aspects for many years. Information on geometrical form of particular anatomic structures should not be underestimated since it is both basic and key information in many clinical cases, beginning from fractures to radiotherapy. Technical apparatuses and devices used in doctors' practice also require comprehension of technical documentation in necessary range. The knowledge of some geometrical issues can help in doctors' work although it is necessary to a certain group of doctors.

Keywords: geometry, engineer graphic, medicine

1 Historical background

Medicine has always developed basing on new technological and technical solutions. This medical knowledge, gained sometimes with difficulty, had to be recorded and the easiest way of

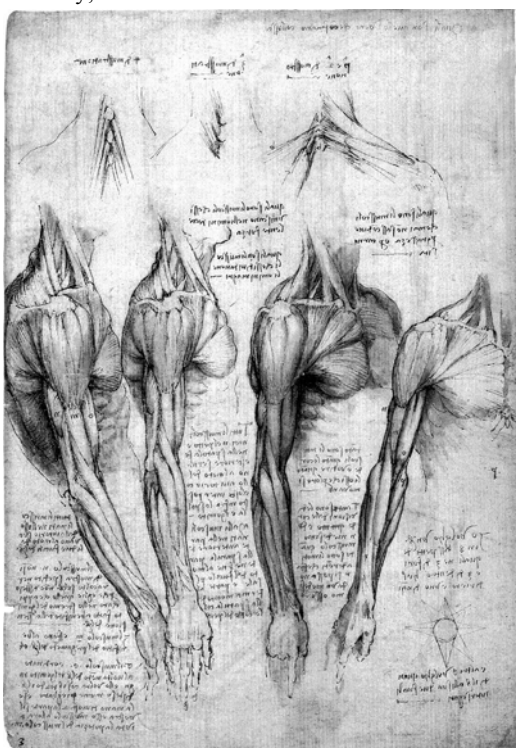


Fig. 1 Leonardo Da Vinci sketches

recording was description in a form of a text and a drawing. In nature it is very rare to come across exactly the same animals or plants and thus it can be even stated that they do not exist. This remark refers to humans too. Therefore, the oral description can

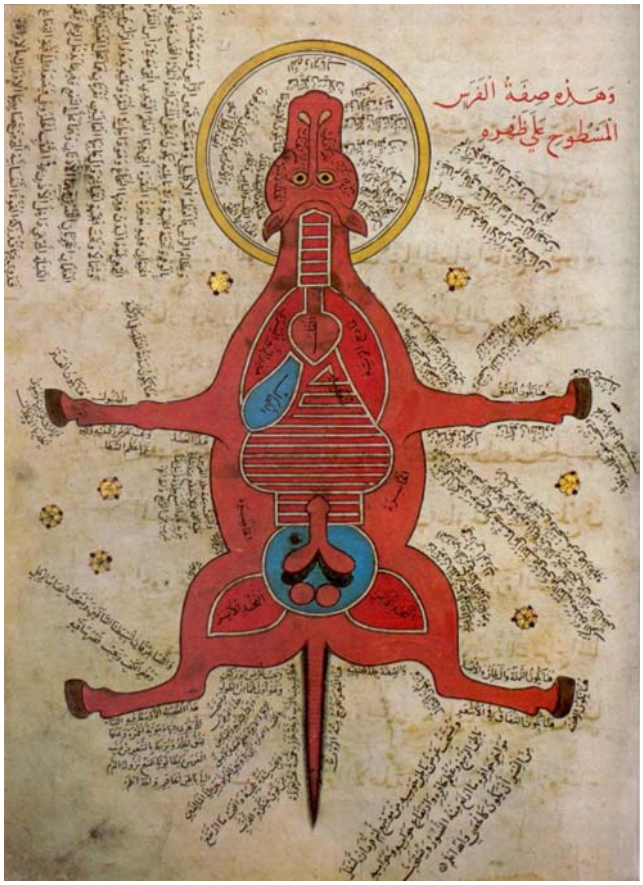


Fig. 2 15th century Egyptian anatomy of horse

refer to the description of symptoms and the way of clinical conduct. It is difficult to imagine the description of e.g. spinal bones and so, since the very beginning of medicine as science, picture is used for presenting typical shape of organs (bones, circulatory system etc.). In many regions of the world and in many historical epochs this method of reaching knowledge was opposed by authority, in many cases caused by religious motives as well. Many scientists and artists gained knowledge on inner organs exposing themselves to danger. The share of painters and sculptors in the development of that method should be stressed. As it can be seen in Leonardo Da Vinci sketches (Fig.1), such a precision could have been made only based on autopsy.

Fig. 3 Acupuncture chart from Hua Shou (Ming Dynasty)

Sketches with such precision could not have been made without with genius of the author, especially when autopsy could be done only at night. Graphical record of gathered information does not refer solely to European culture but also to other cultures e.g. Arabic anatomy of a horse (Fig.2) and Acupuncture chart from

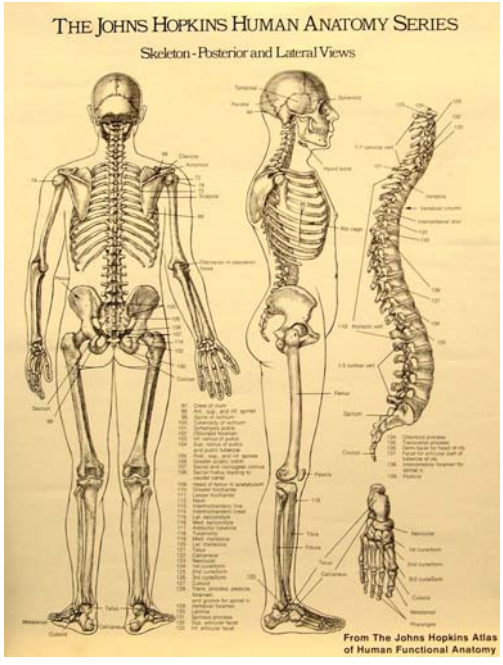


Fig. 4 The Johns Hopkins Atlas of Human Anatomy

Hua Shou (fl. 1340s, Ming Dynasty) (Fig.3). Anatomy atlases have always been made with greatest precision. Since organs are located in some 3D space, the way inner organs were presented varies. The atlas must have been prepared in such a way that a person without special preparation could use it. It was true specially when doctors and doctors-to-be were not prepared as far as geometry was concerned. One of the ways of presenting which was used in many atlases was to show consecutive layers with organs as well as picturing from different sides e.g. front and rear (Fig.4) or models in scale or natural size were made.

2 Processed picture as a base for doctor's conduct in real time

In case of classical operation of inner organs a doctor must first have access to the place which is to be operated on, so that it can be seen and tools can be used freely. Due to these actions big wounds appear. A doctor can directly see operating area and thanks to that he can see some incorrectness or damages which were not included in the process of procedure. Currently a doctor can carry operation without such wounds but a drawback of such solution is the fact that he bases his decisions on a picture obtained by intermediary devices (e.g. camera- screen) and not on what he actually sees. The picture can be distorted owing to optical intermediary devices and therefore knowing about that one can evaluate the situation correctly and carry the operation. Consequently, such practices require training connected with the skills of operating a device which is controlled by man and is not directly in his hand. Similarly, the same situation happens when distant operations are carried and problems appear connected with constant uninterrupted communication between a doctor, operating theatre and a patient.

3 Picture and diagnostics

Another issue worth discussing is interpretation of obtained graphical information from X-rays via ultrasonography and Computed Tomography (*CT*) and finally magnetic resonance

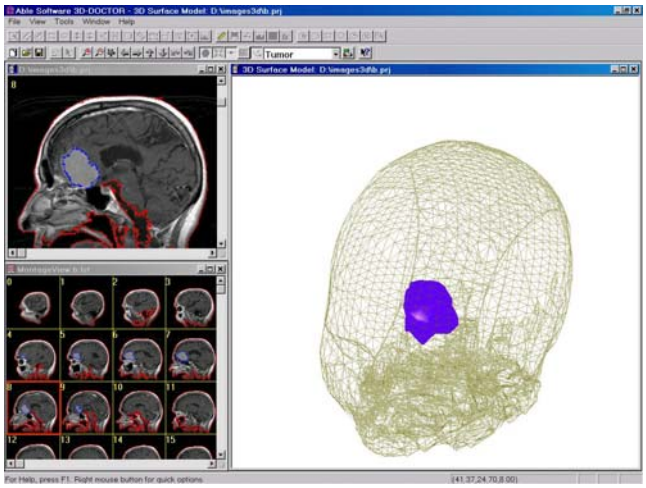


Fig. 5 Skull - 3D model

imaging (***MRI***) (Fig.5). Each of these methods despite engineers' efforts requires interpretation of picture. This interpretation is made by reading geometrical form of a given anatomic structure based on medical knowledge. Computer with its software facilitate the task of creating 3D model of a treated organ greatly. However, each method can be error burdened or have some inaccuracies, which may result from the way information is

collected or algorithms generating 3D model as well as the way of model representation. The engineers' task is to prepare the process in such a way that a model is as close to natural organ as possible. In preplanned processes there is time and possibilities for engineers to check and correct the model so that it is useful for a doctor. Nevertheless, in emergency cases there can be no time for checking and verification. This is the main reason why a doctor who knows at least the outline of model creation and information collection about organ's geometry, can in case of errors (resulting from information collection or model generation algorithm) manage himself.
Creation of 3D model can lead to diagnosis of future complications or susceptibility to injuries e.g. fractures. Such a model can be used for research on e.g. strength [Rychlik et. al. 2004] and verification of different theories.
For the sake of cognitive research the whole library of human body sections have been created with 1mm precision, which can be used for 3D model creation in didactics.

4 3D space in doctors' practice

In some situations operations have been made based on documentations prepared on the basis of 3D model, which truly

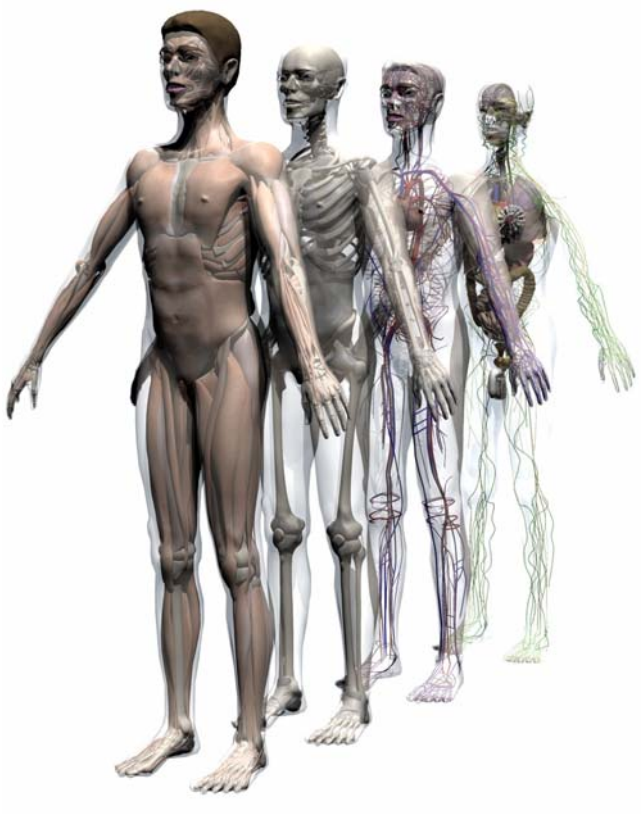


Fig. 5 3D models

represented a patient. Simulation of the way tool is used is necessary especially in case of operations which require great precision e.g. scull operations when tiny error or inaccuracy can lead to damages of brain which are difficult to predict. Preparation of such operation requires first preparation of data for precise creation of 3D model (Fig. 5). Next problem which engineers have to face is preparation of such software which will allow a doctor to get to know operation area. Operation theatre has its own rules and thus everything should be prepared and foreseen. Firstly, a doctor or his assistant must have a possibility to change the location of the observed model and tools in a possibly simple way. Static picture does not guarantee the full control of tool's location. Before operation patient must be precisely positioned as regards tools. It is also necessary to secure patient against relocation or rotation before operation so that coordinates of a patient and tool do not differ.
Another area where the knowledge of geometry is very important are techniques connected with radiation [Wagner et al. 2002]. In this case engineers can help in the possible correctness of results. Also in this situation a patient must be placed in precisely determined place so that a beam of radiation gets to the planned area. In order to reduce side effects of radiation of neighboring tissues the source of radiation must be in motion and it must be adjusted in a way that proper effect is achieved. In this case we talk about geometry and time in 3D space. Similar situation is with teeth X-rays [Stachel 1998] panoramic views.

5 Doctor - picture - patient

In many patient- doctor contacts it is necessary to explain to a

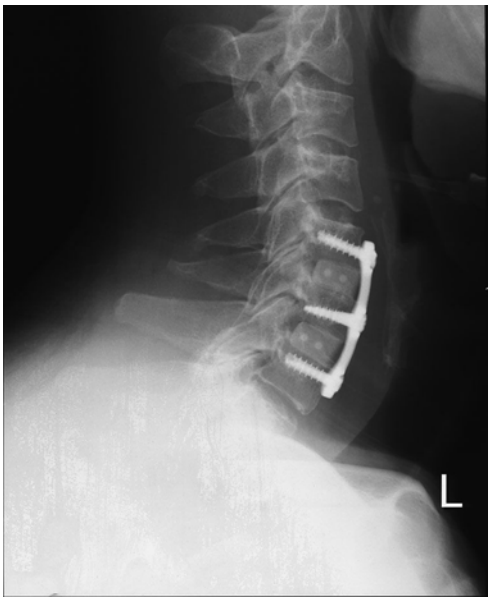


Fig. 6 X-ray vertebral column

patient what his procedure will be like. Naturally, already existing illustrations concerning similar cases can be used. However, in

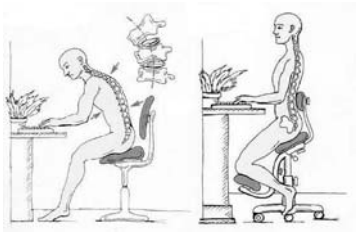


Fig. 7 Sitting position - vertebral column

case of e.g. fractions each case is different and thus the way of joining is different. Detailed information for patients is also important during rehabilitation. When a patient knows what e.g. stiff spine is (Fig.6,7) he can undergo proper rehabilitation easier. Based on 3D models didactic movies for patients and people under rehabilitation can be made.

6 Simulators

The development of technology enables the usage of different simulators, which based on computer systems generate pictures that can be adapted for one viewer (e.g. such a system with goggles as an element). Then generated picture can be exactly the same as in reality. Different situation appears when there are a few people and one picture for them. It is impossible to have it shown correctly for all viewers and therefore being aware of the distortions it is easier to take part in simulation. This remark refers also to presentations with 3D techniques when being aware of possible distortions observer can understand the picture better. Another aspect is the fact that such distorted picture can lead to faster tiredness of viewer, and in consequence incorrect decisions. The error is then not the effect of lack of knowledge but the result of used simulator.

7 What to teach the students of medicine

The important information at Polish medical universities does not teach the students of geometry, computer graphic.
Topics should differ from regular geometry course which is offered at the technical universities, definitely it should include:

- reading of technical documentation
- axonometry (3D models)
- central projection (also from cameras)
- methods of generation of 3D models
- generation of digital picture

8 Conclusions

There are a number of prerequisites to introduce geometry to doctors' education and training.
The following issues should be considered:

- the range of topics depending on the specialization
- the way of realization

9 Acknowledgement

I would like to express my gratitude to MD, PhD Wiesław Rycerski from Upper Silesian Rehabilitation Center "Repty" in Tarnowskie Góry, Łukasz Talarek from Municipal Hospital no 1 in Gliwice and Stanisław Sulwiński, PhD for all the inspiring discussions and MSc Barbara Skarka for help in preparing English version of the paper.

10 About Author

Krzysztof T. Tytkowski, PhD, Eng. is a adjunct (teaching and research) in Geometry and Engineering Graphics Centre Silesian University of Technology. His research interests are computer aided design, descriptive and projective geometry, visualisation, space modelling, 3D reconstruction, visualisation.
Member of: Polish Society of Mechanical Engineers and Technicians (since 1987), Polish Society for Geometry and Engineering Graphics (founder, since 1994), International Society for Geometry and Graphics (since 1997), Polish Association for Promoting Computer Engineering Systems ProCAx (founder, since 2000).
He can be reached by e-mail: Krzysztof.Tytkowski@polsl.pl, by fax: +48 32 2372658 or through postal address: Ośrodek Geometrii i Grafiki Inżynierskiej – RJM-4/ Politechnika Śląka/ ul. Krzywoustego 7/ 44-100 Gliwice/ Poland

11 References

RYCHLIK, M., MORZYŃSKI, M., NOWAK, M., STANKIEWICZ, W., ŁODYGOWSKI, T., OGURKOWSKA M. B., 2004, Acquisition and transformation of biomedical objects to CAD systems, Strojnicky Casopis, No. 3/04 :121-135. Bratysława

STACHEL, H. 1998 New Applications of Geometry, Journal for Geometry and Graphics, Vol. 2, No. 2, pp. 151 - 159

WAGNER, D., WEGENKITTL, R., GRÖLLER, M.E., 2002, EndoView: A Phantom Study of a Tracked Virtual Bronchoscopy, Journal of WSCG, 10(2):493-498

Fragments from the Swedish history of computer graphics with SIGRAD

Lars Kjelldahl, CSC, KTH,
lassekj@csc.kth.se

Abstract

This overview is to some extent fragmental, but may hopefully give some patterns of the development of the computer graphics area in Sweden, from the horizon of SIGRAD, the Swedish Computer Graphics Society. It gives some facts of 30 years of computer graphics from a Swedish perspective. Things discussed include early software, applications, people involved, international visitors, computer graphics societies in other countries, conferences, bulletins/newsletters, courses and seminars.

CR Categories:

Keywords: history, evolution, computer graphics

Foundation of SIGRAD

The first pictures drawn with a computer in Sweden was performed on the BESK computer around 1953 on an oscilloscope. Not very much happened after that, but during 70ies the interest in the possibilities to draw pictures with computers grew at several places. Some expensive displays (vector technology) were bought, different plotters were used including plots done at ordinary line printers.

The time was mature to collect people around the possibility of drawing pictures and the call for an association gave responses from around 100 persons, which was a surprise for some people. At a meeting with 55 participants at KTH in December 1976 the Swedish Computer Graphics Association was founded, later given the name SIGRAD.

The people that showed interest in the new association came from different disciplines with different backgrounds having interest in areas such as basic software, CAD/CAM, GIS, computer art, statistics, and industrial supervision. This turned out to characterize and to be a strength for SIGRAD. Due to the variety in interest it was also decided to keep the new society independent to other organizations. Things as standardization and terminology were pointed out as being of special interest for the society. It was also expected to be a forum for discussions and dissemination of information. Financial support from various sources in the beginning made it possible to work without member fee during the first five years when building the infra structure.

The first interim executive committee had members from KTH, SCB, Lunds Datacentral, Stockholms Datacentral, Elektronmusikstudion/EMS, SAAB-Scania, Statens Vägverk, K-konsult and HSB. After six months a list of 140 names with interested people had been collected. The application areas represented varied very much, which contributed to the society becoming a very dynamic forum.

The first years focused on two topics: to show the variety of applications and to discuss the variety of software with a need for standardization.

A seminar conference in April 1977 included presentation of 7 different software packages and 100 participants came to this event.

A similar seminar but on graphics displays was held in November, 1977. This year the first bye-laws were presented by Axel Andersson, who also suggested the name SIGRAD for the new society.

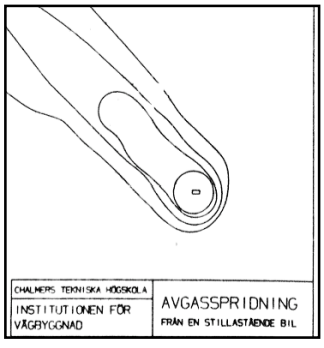
Kind of activities

The activities of SIGRAD have included annual conferences/events, courses, education workshops, study visits, international cooperation with societies and experts from various countries mainly in Europe, publication of a bulletin/newsletter, maintaining a web page (when the use of www had become established) and keeping a list of individual members with annual fees.

Application areas

The applications that have been presented during SIGRAD events, study visits and conferences varies. Some applications have after some years been an established part of day to day work and therefore became of less interest for a society like SIGRAD. One example of such an area is business graphics. Applications include CAD, architecture, GIS, HCI, mathematics, statistics, environmental visualization, games, chemistry, medicine, virtual reality, business graphics etc

Below are included examples of early applications such as presented in a few images from a special bulletin issue from 1979.



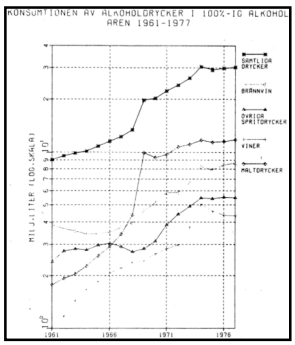


Figure 1: Illustrations of pollution distribution and alcohol consumption

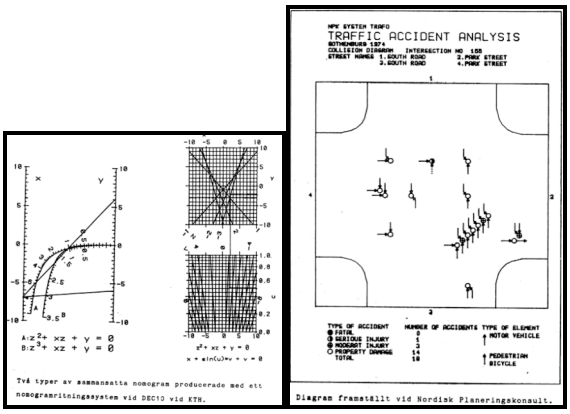


Figure 2: nomogram drawing and traffic analysis

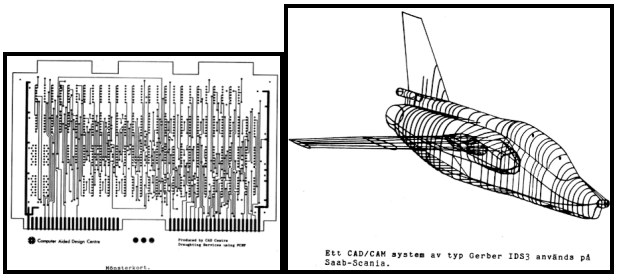


Figure 3: printed circuit card and CAD/CAM picture

Software

In the middle of the 70-ties a numerous variety of software packages existed. These packages were often adapted to different hardware concepts and to different applications. Due to this variation the packages used basic graphics concepts that were incompatible and often very hard to convert. It was also hard for programmers and users who had to relearn when switching from one package to another. The international efforts also had seen these problem and they were discussed at the Seillac workshop in 1976 [1], which later gave the result when an international standard called GKS was accepted by ISO. The standard was used during a few years but was later in practice abandoned for software adjusted to the latest technology.

The SIGRAD events during the 70-ties often had a focus on basic computer graphics software, After that, when de facto standards driven by technology dominated, SIGRAD spent much less efforts in basic software problems.

Hardware

The hardware development were important in the 70-ties and 80-ties. The interest were focused on three sub areas, i.e. displays, plotters and interaction devices.

Conferences

The annual conference was established in 1981, but also before that there were some one day events with a flavor of a conference. In the appendix of this paper a list of notes about the conferences/events can be found.

One can see some trends in these notes. In the beginning software and to some extent hardware was in focus together with applications. Later when the standards were established the interest shifted towards fundamental techniques and concepts such as interaction, colour, text and image etc Later when web appeared web graphics together with visualization captured the interest of the events. From the beginning of 2000-ties there was a shift from an event with invited speakers towards a more scientific event with submissions and a review process. The proceedings then also become a publication that was printed and made available through web (links are given at the end of the paper). Finally it should be said that applications have always been an important part of the activities.

Courses

In 1979 a two day course in computer graphics with a focus on software for colour printers was given in cooperation between Lund data center and SIGRAD. The software used was later developed to the UNIRAS software. The interest for the course forced the organizers to give the course twice.

As part of the annual conference some basic courses were also occasionally given. International experts have sometimes been used, such as José Encarnaçâo (a course on standards for computer graphics) and Daniel Thalman (a course on animation).

Study visits

As part of the annual conference or as part of the annual assembly there have often been study visits at different companies. Examples of companies and institutions that SIGRAD visited are IBM, Silicon Graphics, SUN, Sjöland&Thyselius, FOA, KTH, Boliden, Chalmers, SVT (public television), HSB, PCG, ABB These visits have usually had 10-30 attendees with many discussions and contacts being established.

Education workshops

At regular basis SIGRAD arranged education meetings with discussions on course curriculums, lab assignments, text books etc. The meetings were usually held biannually with 10-15 participants and at different universities in Sweden.

Bulletin and newsletter

An important forum for the SIGRAD was the ”SIGRAD Bulletin”, who was published 2-3 times/year. These bulletins provide an interesting reading about the society from 1976-1997. After 1997 a newsletter replaced the bulletin. It was sent out by email and was made available through the web site of SIGRAD [2].

An interesting bulletin is for instance, the one published in January 1979 with collections of pictures from different applications (mentioned earlier).

Cooperation with other societies, international visitors

Early, during 70-ties, a contact was established with our sister organization in Norway, Norsigd being started approximately at the same time as SIGRAD (actually two years before SIGRAD).

Norsigd had/have a somewhat different character than SIGRAD using company members instead of individual members, and also getting incomes from a software product (GPGS-F). En early contact with Norway also resulted in a two week Nordic Research course on computer graphics in Trondheim, 1978, organized mutually between people from Sweden and Norway.

Contacts between Eurographics and SIGGRAPH have existed for many years. An affiliation agreement with Eurographics was signed in the 90-ties. In 2007 this agreement was replaced by SIGRAD becoming a Eurographics Chapter.

The contacts with SIGGRAPH have resulted in some agreements on discounts etc.

SIGRAD has during the years tried to get international speakers for conferences and courses. Some examples of people that have visited Sweden usually invited by SIGRAD are Daniel Thalman, José Encanação, Jos Stam, David Duce, Judy Brown, Dale Sutcliffe, Ian Hurrington, Roger Hubbold, Martin Göbel and Andries van Dam.

People involved

Many different kind of people have been involved in SIGRAD, some for just a year and some for decades. We list some of the key persons below and start with the chair persons:

Lars Kjelldahl, 1976-1980, Sten Hultman, 1980-1981, Mikael Jern, 1981-1985 , Anna Holst, 1985-1990, Lars Kjelldahl, 1990-1993, Olov Fahlander, 1993-1995, Mikael Jern, 1995-2001, Anders Ynnerman, 2001-2004, Anders Backman, 2004-2007, Kai-Mikael Jää-Aro, 2007-

A selection of other persons: Sten Kallin, Jan Lidén, Ulf Rozén, Brita Larson, Michael Pääbo, Pierre Lingheim, Kersin Malmqvist, Gunnar Petersson, Sven-Ove Westberg, Larsgunnar Nilsson, Magnus Bondesson.

Concluding remarks

In addition to issues discussed in the previous paragraphs, it is interesting to observe the shift from Swedish as main language for the activities (with lists of Swedish terms in computer graphics) towards more English orientation with the annual conference also becoming more international from that point of view.

A project to describe the Swedish IT history is currently taking place (http://ithistoria.se/). This paper could be seen as a small contribution to that work.

SIGRAD conferences, list of short notes

Miniconference, April, 1977, KTH: software presented was GINO-F, GPGS-F, GCS, GD3, DISSPLA, COLOR, PLAM
Miniconference, April, 1978, KTH: GIS, with speakers from Lunds Tekniska högskola, KTH/fotogrammertri, Statens lantmäteriverk, Jacobson&Widmark, K-konsult, VIAK and Centralnämnden för fastighetsdata
Miniconference, March, 1980, Skellefteå: CAD-systems, local activity

SIGRAD81: December, Esso Motor Hotel, Järva Krog: Survey of computer graphics with a focus on software and the experience of the use of software.
SIGRAD82: December, Stockholm, Interaction, Colour and Ergonomy
SIGRAD83: Trygg-Hansa, Stockholm, Picture/image output, transfer, and storage. Among speakers: Mikael Jern, Yngve Sundblad, Program committee: Mikael Jern, Lars Kjelldahl, Brita Larson, Anna Holst
Course, March, 1984, Andries van Dam, together with Paralog, two day course
Course, 1984, José Encarnação, GKS, survey of computer graphics
SIGRAD84: December, Stockholm Convention Center, Program committee: Sten Kallin, Lars Kjelldahl, Anna Holst. Theme: text and image, Among speakers: David Duce, Pete Harrison, Jan Winblad, Jan Johansson, Jerker Lundequist, Astrid Sampe, Veine Johansson, Staffan Romberger, Sten Carlqvist
SIGRAD85: December, Stockholm Convention Center, two one day courses and a two day conference on computer graphics with state of the art and trends, Among speakers: Dale Sutcliffe, Two courses were given (tutorial on computer graphics; standard introduction)
SIGRAD86: Inhouse publishing, Silja Line, conference on the boat with a trip between Stockhom and Helsinki (and back)
SIGRAD87: Computer Graphics with a message – image and text, Silja Line, conference on the boat with a trip between Stockhom and Åbo (and back)
SIGRAD88: Kiruna, Interaction and GIS
SIGRAD89: Electrum, Kista, Object oriented methods, architectural applications, interaction
SIGRAD90: IBM Forum, VR – computer graphics is changing
SIGRAD91: SUN, Kista, Graphical User Interfaces
SIGRAD92: KTH, Computer graphics and planning of resources
SIGRAD93: Linköping, animation course, interest groups (VR, education, art, animation, etc)
SIGRAD94: Nacka Forum, Living pictures, visualization, with among others Judy Brown and Paul Rea
SIGRAD95: Nacka Forum, Visualization, animation and interaction in Windows, invited speakers were Martin Göbel and Ian Currington
SIGRAD96: Nacka Forum, web graphics, including talks on Spotfire and AVS/UNIRAS
SIGRAD97: Graphics on web, Nacka Forum, Stockholm
SIGRAD98: 3D Visualization
SIGRAD99: Stockholm, KTH, 3D Graphics on the Net Facing 2000, talks on CAVE,
SIGRAD2000: Norrköping,
SIGRAD2001: Stockholm, KTH, Real-Time Graphics
SIGRAD2002: Norrköping, Special Effects and Rendering [4]
SIGRAD2003: Umeå, Real-Time Simulations [5]
SIGRAD2004: Gävle, Environmental Visualization [6]
SIGRAD2005: Lund, Mobile graphics [7]
SIGRAD2006: Skövde, Computer Games [8]
SIGRAD2007: Uppsala, Computer graphics in healthcare

References

[1] Guedj, Tucker, Methodoly in computer graphics, North-Holland, 1979
[2] www.sigrad.org, www.sigrad.se
[3] conference documentations (paper, 1981-2001)
[4] www.ep.liu.se/ecp/007, 2002
[5] www.ep.liu.se/ecp/010, 2003
[6] www.ep.liu.se/ecp/013, 2004
[7] www.ep.liu.se/ecp/016, 2005
[8] www.ep.liu.se/ecp/019, 2006

Graphical Literacy Development using Learning Management System

Zoja Veide [*]
Department of Computer Aided Engineering Graphics
Riga Technical University

Veronika Strozheva [†]
Department of Computer Aided Engineering Graphics
Riga Technical University

Abstract

In the given article the example of learning management system is presented. The learning management system was used for study of compulsory subject of Descriptive Geometry and Engineering Graphics for students of extramural department of the Riga Technical University (RTU). In addition the Blackboard Learning System (BB) was used for accommodation of a teaching material. The general structure of materials placed in environment of the BB system includes: theoretical material, performance of training exercises, and performance of the tests. Results of the research presented in this paper give possibility to make conclusions on efficiency of use of the above-mentioned systems in mastering of a subject of Descriptive Geometry and Engineering Graphics. The article is a continuation of the research [Veide and Strozheva 2007], presented at Vth Conference Geometry and graphics, and covers the period from September 2006 till June 2007.

Keywords: engineering graphics, e-learning, Blackboard Learning System

1 Introduction

The use of modern information technologies in training (network distance training; network distance courseware; virtual universities; etc.) opens new opportunities for development of the system of education [Boyle et al. 2005]. However, the effective introduction of information technologies in training is connected to adaptation of the participants of educational process to information technologies. In addition opportunities of computer engineering and program systems must be adapted to the purposes, tasks and at last to participants of educational process. Recently, a variety of schemes of distance learning has arisen. These schemes use electronic ways of linking between the learner and the source of instruction with increased interaction between them.

[*] zv@neolain.lv
[†] vs@pit.lv

Distance learning (DL) is a strategy developed to harness the power of learning, information, and communication technologies to modernize education and training [Manning 2000]. The DL initiative is intended to implement the "anytime-anywhere" learning concept to provide access to the highest quality education and training that can be tailored to individual needs and delivered cost-effectively, whenever and wherever it is required.

The DL is structured learning process without the physical presence of the instructor. The DL is enhanced with the technology. It may draw upon resources which are physically distant from the location where learning is taking place [Jonsson 2005].

Now standards of learning technology use the learning management systems which include new functionalities and capabilities such as back-end connections to other information systems, complex tracking and reporting, centralized registration, on-line collaboration and adaptive content delivery. The Blackboard Learning System was used for improvement of quality of educational process in the Riga Technical University.

The BB environment delivers a course management system, customizable institution-wide portals, online communities, and an advanced architecture that allows for Web-based integration with administrative systems. The BB system is a kind of software applications specially designed to enhance teaching and learning. Intuitive and easy-to-use for instructors, the Blackboard Learning System is built on a scalable enterprise technology foundation that facilitates growth and performance. Institutions around the world use the Blackboard Learning System to:

- Create powerful learning content using a variety of Web-based tools;
- Develop custom learning paths for individual students or groups;
- Facilitate student participation, communication, and collaboration;
- Evaluate students' work using a rich set of assessment capabilities;
- Bring top publisher content into e-Learning.

Only a browser is necessary for students who are using the BB system. Most students in Latvia have personal computers with fast internet connectivity. Those who do not have computers can access them in libraries or in computer classroom of the RTU. Hence instructors may assume that everybody is able to practice using the e-materials offered. Students can access systems like the BB system using their hand-held devices, e.g. the new Nokia E61 cellular telephone [Caprotti et al. 2002]. This mobile phone has a

full keyboard and the conventional browser software that runs on it and this gadget can be connected to the internet. Browsing such hand-held devices, which can be expected to be very common in the near future, will also make real the use of automatically graded quizzes and examinations in any class room.

2 Course in BB environment

The duration of a bachelor program at Civil engineering Department in the RTU is 7 semesters and each of them consists of 16 weeks and additional 4 weeks of exam session. The compulsory subject of Descriptive Geometry and Engineering Graphics is limited to 3 ECPS and students must learn it in the first semester. In this course the students have to complete all the individual home assignments and final exam.

The learning material of the course of Descriptive Geometry and Engineering Graphics has been placed in the BB environment as an experimental prototype to help students in their mastering of topics of the learning subject. The given subject traditionally is difficult for study and mastering, in particular for students of extramural department. The use of the BB system was not obligatory for the students. In addition the students have had possibilities to use the BB environment with an additional opportunity to study theoretical material, as well to get consultations with an instructor and perform their exercises in order to master the learning subject of Descriptive Geometry and Engineering Graphics.

The material of the course of Descriptive Geometry and Engineering Graphics in the BB environment includes [Stanchev 2001]:

- A brief description of the course and a table of contents;
- Announcements and important information for participants;
- Instructor's e-mail address, numbers of telephone and fax, and regular mailing address including office hours;
- The themes of theoretical material of the course including goals and objectives;
- A list of all exercises, assignments and other tasks of the course;
- Explicit information on how students will be graded on assignments, tests, participation; each assignment should be linked to relevant course documents;
- A list of supplementary books, and other learning resources;
- A possibility of interactions with the instructor by means of e-mail communication;
- An opportunity of discussions between participants of educational process.

The students become more competent in formulation of their questions when they use e-mail for dialogue with the instructor. Hence, to ask the question, the student must understand the topic. Necessity to ask questions and to discuss them induces the students for preliminary independent work with a material of the learning subject.

Participants of the BB system carry out exercises and tests placed in the BB environment before a final exam. Exercises, which are placed after corresponding themes of the course, are offered to be performed by the students for their becoming more self-prepared for the tests. Using the BB system the students perform their exercises without limitation of the time and it gives them possibilities to repeat their attempts to solve the tasks. In result the information on the earned points and the right answers are accessible to the students. Thus, the students can independently estimate their level of self-readiness for performance of the tests. The tests in the BB environment are limited with the time and there is only one opportunity for the students to perform the tests correctly. Only final points are accessible to the students. During the distance course of education, the BB system assesses the students formally. It indicates a level of mastering of the course by the students and the standard attained. The formal assessment is intended to verify how well students are able to meet the session objectives on the one hand, and a depth of knowledge learned, on the other hand.

3 Feedback from students

The students, who perform their exercises and tests placed in the BB environment, develop their skills in solving the tasks of the subject learned and accordingly pass their final exam more successfully than those students, who did not use the system of BB (Fig. 1). The results of final exams show that the most of students, who learned the subject in the BB environment, got 7-8 points. The students, who not learn it in the BB system, got 5-6 points.

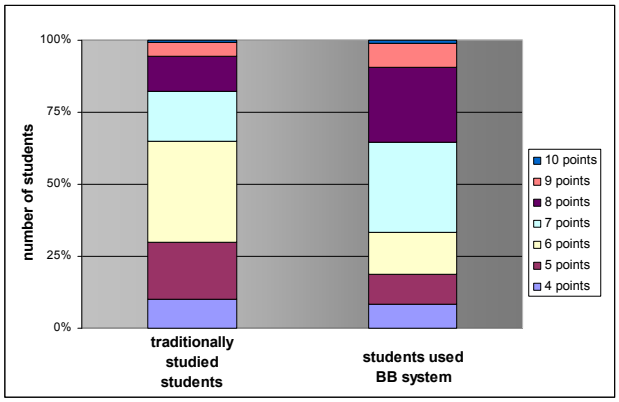


Figure 1: Results of final exams of the course of Descriptive Geometry and Engineering Graphics.

Comparison of results of the final exams of the students, who learned the subject in the BB environment, and those, who did not learn it in the BB system, gives the opportunity to make conclusions on an efficiency of programs similar to the BB programs, which are used for improvement of quality of training [Moreno et al. 2006]. It also is interesting to note that a frequency of visits of consultations, which arise in educational process, has become decreased, but in spite of that the quantity of the questions remained the same the questions itself have become more concrete.

The students demonstrated the best results under the tests placed in the BB environment in following themes: Fundamental views - Point, Orthographic projection and Drawing procedure. Less successful results of the tests were got by the students on such themes as Line, Plane, Sections and Intersections. These results of the tests give possibility to make a conclusion that lectures and

graphic works, which the students must carry out independently, are necessary for mastering of the themes of the descriptive geometry, connected with methods of construction.

The use of the BB system is a free choice of the students. So it makes an opportunity to observe a degree of interest of the students to use this method of training, to reveal main problem of topics learned and to master necessary skills. The term of registration for persons interested to use the BB system has been limited with one month. Thus, a group of participants, which consists of 73 people, has been generated during the limited time of registration.

The students chose intensity of studying of the learning materials independently. Dependence of the intensity of the work in the BB environment during the semester is shown on Fig. 2. The students have shown their greatest activity immediately after registration in the BB system. It makes an opportunity to display the difference between a level of students' interest to a new method of training in a period of time immediately after registration in the BB system and, traditionally, the same their interest at the end of the term of the training. Frequency of use by students of areas of the main content in the learning course of Descriptive Geometry and Engineering Graphics in the BB environment is shown on Fig. 3. The results of final exams show that higher estimates were received by the students who used the BB system more actively.

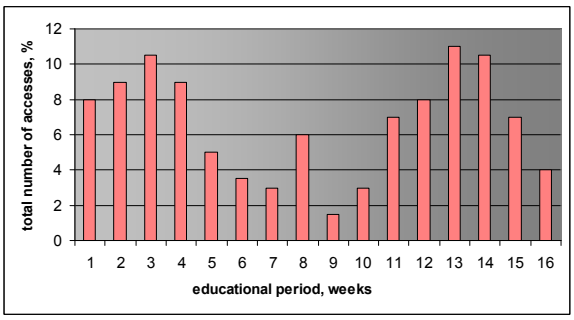


Figure 2: Correlation between activity of the students in the BB environment and duration of educational period

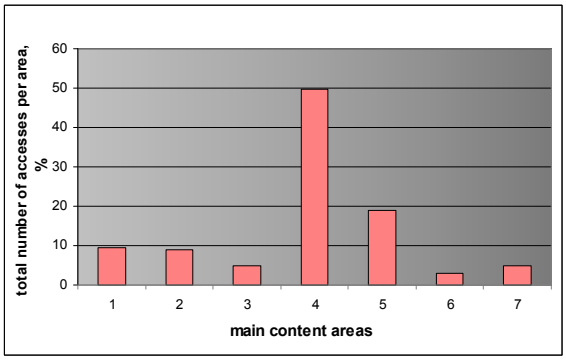


Figure 3: Overall summary of usage by students the areas of the main content in the BB course of Descriptive Geometry and Engineering Graphics. 1 – announcements and important information for participants; 2 – the brief description of the course; 3 – the staff information; 4 – the themes of theoretical material of the learning course including goals and objectives; 5 – tests; 6 – a list of supplementary books, and other learning resources; 7 – communication area.

An anonymous evaluation was carried out at the Riga Technical University in the end of the learning course of Descriptive Geometry and Engineering Graphics. The students' answers to a question "Why results of the BB tests are insufficiently good?" are shown on Fig. 4. This question was addressed to the students who have received an average estimation below 6 as result of the tests. Training exercises have been placed in the BB environment for improvement of preparation of the students to the tests.

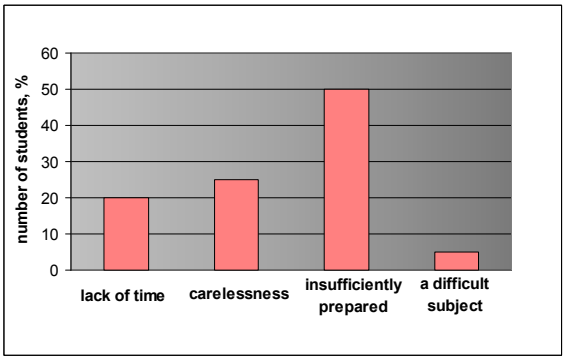


Figure 4: Students' answers to a question "Why results of the BB tests are insufficiently good?"

All the students whose responses are in this survey told that they would recommend the learning course to their fellow students. Since the assessment was conducted anonymously, the responses reflect students' perceptions of the learning course. The students' feedback was very positive, and it is clear that the learning course of Descriptive Geometry and Engineering Graphics in the BB environment will emerge as a real option at the university level.

4 Conclusions

Our experiences in using the BB system in actual teaching process have confirmed the potential of the BB environment as a beneficial and useful system for teaching. The system enables the user to produce electronic materials for the learning course simply as a by-product of teaching.

The results of the final tests of the learning course of Descriptive Geometry and Engineering Graphics, which are presented in this article, demonstrate that higher average estimate (7,4 points) were received by the students who used the BB system. The students, who not learn it in the BB system, got average estimate 6,2 points. To explain this fact it is possible to suppose that the students, who use the BB system, have an opportunity to carry out more exercises and tests in the learning course. Therefore, the quality of the training course, that is necessary for development of graphic literacy of engineers, improves.

The main challenges in educational system are how to keep students focused in the subject matter, how to motivate them to work independently, and how to fight the high rate of the students' dropping out. These challenges are accentuated in the on-line setting in which the direct contacts between the students

and the instructor are limited. It is easier for the students not to focus when the instructor does not see them. And it is easier for them to drop out because, in the on-line setting, they interact with a computer and not with a person [Caprotti et al. 2002].

Like the development of good textbooks, the development of good educational software is a long-term process of trial and error that will need continuously to draw on the experience of the best tutors, those who are responding to the needs of individual learners. Since education is the main pillar of our society's future, all efforts in that direction are well spent.

References

Boyle, T., Bradley, C., Chalk, P., Fisher K., Pickard, P., 2005. Introducing a virtual learning environment and learning objects into higher education courses. *International Journal of Learning Technology, Vol. 1, No.4,* 383 – 398.

Caprotti, O., Seppala, M., Xambo, S., 2002. Novel Aspects of the Use of ICT in Mathematics Education. http://webalt.math.helsinki.fi/content/e16/e301/e818/finalCISSE2006--pdffile_eng.pdf

Jonsson A., 2005. A case study of successful e-learning: A web-based distance course in medical physics held for school teachers of the upper secondary level *Journal "Medical Engineering & Physics", 27,* 571–581.

Manning, L., 2000, Advanced Distributed Learning. *DISAM Journal, Summer.* http://findarticles.com/p/articles/mi_m0IAJ/is_4_22/ai_67318433

Moreno, L., Gonzalez, C., Castilla, I., Gonzalez, E., Sigut, J. Applying a constructivist and collaborative methodological approach in engineering education. *Journal "Computers & Education",* article in press, available online 14 February 2006.

Stanchev, P., 2001. Distance education courses. *International Journal „Information Theories and Applications", Vol.8, No.1,* 41 – 46.

Veide, Z., Strozheva, V. 2007 Application and efficiency assessment of e-learning software. In *Proccedings of* ***V****th Conference Geometry and Graphics, Ustron 2007,* article in press.

Realistic Virtual Characters in Treatments for Psychological Disorders An Extensive Agent Architecture

Anja Johansson*
Dept. of Science and Technology
Linköping University
60174 Norrköping, Sweden

Pierangelo Dell'Acqua†
Dept. of Science and Technology
Linköping University
60174 Norrköping, Sweden

Abstract

Interactive virtual reality applications have lately been used as a complementary tool in the treatment of people with different kinds of psychological disorders. A number of these projects would benefit from a more realistic behavior of virtual characters. In this paper we describe our ongoing work on an advanced emotional agent architecture for virtual characters and propose new potential applications within the area of health care.

Keywords: Virtual reality, artificial intelligence, agent architectures.

1 Background

Computer animation is concerned with producing sequences of images that when displayed at high speed give the illusion that certain components of the image move. The aim is to produce animations that look realistic. Computer animation has typically focused on low-level locomotion problems, i.e., low-level control.

In the mid-1980s, researchers began incorporating physical principles to develop physical models for animating passive objects, such as falling and colliding objects. The idea behind this kind of model was to explicitly represent physical concepts. In the last decades, researchers started to focus on the requirement that characters in animations should behave realistically. That is, characters should be able to perform sequences of movements. This is commonly referred to as high-level control problem. Research in behavioral modelling progressed toward self-animating characters that react appropriately to stimuli perceived from the environment. The seminal work in this area was that of Reynolds [Reynolds 1987]. Tu [Tu 1999] and Terzopoulos [Terzopoulos 1999; Tu and Terzopoulos 1994] extended then that approach to dealing with some complex behaviors. One of the early attempts at virtual autonomous characters was the Improv [Perlin and Goldberg 1996] system. Improv allows for the creation of real-time behavior based animated actors using an animation engine and behavior engine to allow authors to create sophisticated rules governing how actors communicate, change, and make decisions. At the top of the modelling pyramid, cognitive modelling has emerged as the use of artificial intelligence techniques, including knowledge representation, reasoning, and planning, to produce virtual characters with some level of deliberative intelligence, see for example [Funge 1998; Funge 1999; Funge et al. 1999]. Addressing human cognitive functionality is a challenging research area in artificial intelligence (AI). Funge pioneered the use of hardcore artificial intelligence techniques in computer games and animation. His aim was to devise a system suitable for rapid prototyping and producing off-line animations. To do so, he used logical reasoning to shift most of the work for generating behavior from the animator to the animated characters. With the intention that the devised system should be easy to build, reconfigure and extend. In [Funge 1998] Funge addressed the problem of devising characters that display elaborate behavior in unpredictable and complex environments. In order to generate high-level behavior, he used the situation calculus [Reiter 1991] to model the virtual world from the animated character's point of view. The basic idea was that a character viewed its world as a sequence of snapshots known as situations. An understanding of how the world could change from one situation to another could then be given to the character by describing the effect of performing each given action. Since the world was dynamic, characters were equipped with knowledge-producing actions (like sensing). Funge's work contributed to advanced computer animation research in several ways: (i) he was the first to tackle the problem of modelling cognitive capabilities of virtual characters enabling them to represent, reason and act in their virtual world, and to perform sensing actions; (ii) to develop and implement an architecture suitable to specify the high-level control combining the advantages of a reactive and a reasoning system. The devised architecture was such that cognitive models were built on top of biomechanical, physical and kinematic models (that control for example locomotion, collisions, etc.). The interaction among the various levels in the hierarchy is achieved by communicating the actions to be performed to the lower levels where they are executed by the appropriate routines. In contrast, the function of sensing actions is to prompt the lower levels to return information about the state of the world to the cognitive level.

2 Research Vision and Objectives

We consider virtual, autonomous characters situated in dynamic, unpredictable, virtual worlds. To render these characters self-animated (i.e., alive) in real time performances, we need to make them able to perceive, reason and act in the world where they are situated. This goal can be achieved by ascribing characters cognitive capabilities together with reactive functionalities that implement primitive behaviors (such as avoiding collisions). Our aim is to be able to deploy autonomous characters in interactive systems and virtual reality applications. Despite several attempts to make the animated characters more autonomous and intelligent there is still a broad scope of research into the problems involved in the design and modelling of virtual characters. The growing popularity of logic-based agent architectures gives new opportunities to apply recent advances in AI to this problem. Hence, the focus of our project is to deploy state-of-the-art artificial intelligence methods into character simulation and extract and address new research issues in computer graphics and artificial intelligence in the context

*e-mail: anjjo@itn.liu.se
†e-mail: pier@itn.liu.se

of the developed applications.

3 Current Agent Architecture

At present time the architecture (depicted in Fig. 1) includes the handling of emotions, an appraisal module, a simple decision module, a knowledge base and a trust module. The original design and implementation of the architecture was carried out by two diploma works [Esbjörnsson 2007; Johansson 2007]. Our system architecture needs to be modular to simplify the task of adding and editing components. The system is rather extensive and management of such a system is easily done in a modular architecture.

The system works in the following way. At each update time (preferably at not every frame as this will slow down the performance of the entire system), the simulation engine (e.g., a game engine) sends information to the XML interface. Such information may be about the whereabouts of objects, weather conditions etc. In the XML interface the information is converted to XML and sent to the knowledge base (although some specific types of information are sent directly to the Affective Appraisal module). In the knowledge base the information is added to the agent's memory (we use XSB Prolog system to store the data). The Affective Appraisal module triggers emotions and expectations depending on what happens around the agent. Emotions and trust also influence which emotion and expectation to trigger. The decision module selects which action to execute depending on the current knowledge, emotional state and trust. The selected action is sent back to the XML interface to be executed by the simulation engine.

In the following sections we will describe the different modules of our system in more detail.

3.1 XML for Inter-Modules Communication

To keep our agent architecture independent of simulation and rendering engines, we have developed an XML interface between the chosen simulation engine and the agent architecture. Information from the game state, no matter how it is stored there, is converted into an XML representation that the agent can process. This allows us to decouple the chosen simulation engine from the agent architecture in the way that changes in the representational language of the simulation engine are not reflected into the modules.

Communication between the different modules in the agent system is also largely done in XML. This makes modifications of the modules easier. XML is also used as a configuration language to define the settings of the different modules.

To ensure fast XML parsing we use a small DOM-based XML library named TinyXML. Because it maintains the entire XML document in memory there is no need for writing information to file. TinyXML provides ways to quickly traverse XML node trees and even with the extended checks we added to ensure well-formed documents, the system is fast.

3.2 Emotion Module

The original emotion module was developed in a prior diploma work [Esbjörnsson 2007]. The emotions in our system are represented as signals[1]. The set of all signals of the same type forms the corresponding emotional state. Each signal has the form of a sigmoid curve and consists of the following phases: delay, attack, sustain and decay. The sigmoid curve is defined as:

$$sigmoid(t) = \frac{g}{1 + e^{-(t+h)/s}} + v$$

where t is the time, g is the gain, h is the horizontal shift, s is the slope steepness and v is the vertical shift. Being the signals parameterized, it is possible to create fairly diverse types of emotion signals.

Emotions can influence each other through a sophisticated filtering system (see Fig. 2). There are currently three types of filters: sigmoid, linear and gamma filters. They have the following mathematical definitions:

$$sigmoid(x, g, s, h, v) = \frac{g}{1 + e^{-(x+h)/s}} + v$$

$$linear(x, s, v) = x * s + v$$

$$gamma(x, g, s, v) = g * x^{s} + v$$

where x is the input value to the filter, g is gain, h is the horizontal shift, s is the slope steepness and v is the vertical shift. Note that these functions have dynamic parameters that will change over time as the emotional states change. This provides a powerful mechanism for letting the emotional states interact with each other. The emotional states can influence all aspects of incoming signals, including intensity and phase lengths. The filtering system is completely customizable through a configuration file written in XML.

The filtering system consists of modifiers each affecting one parameter of an incoming signal of a particular emotional state. A modifier is a dynamic filter object consisting of one of the previously mentioned filter functions. Each modifier also consists of several arbiters, each affecting one parameter of the filter function of the modifier. To each arbiter, any number of emotional state can be connected with different influence values.

3.3 Knowledge Base

The knowledge base represents the memory of the agent. Here all information on events, expectations, etc. is maintained. The logic language (XSB, or Tabled Prolog [XSB-Prolog 2007]) we use for the knowledge base also allows for expressing reasoning although this is not yet integrated into the system. XSB allows us to store complex information in a simplified way and it also allows us to create simple rules that can be evaluated dynamically. Incoming information (represented in XML within the agent) from the game state is automatically translated into XSB statements that are asserted into the memory.

Here follows an example of how the XML information is translated into Prolog statements. Suppose that the agent receives an event from the simulation engine stating that there is a warrior at position (1.4, 2.3) holding an axe. This is represented in XML as:

```
< object id="2" type="warrior" >
     < position x="1.4" y="2.3" / >
     < holdsitem type="axe" / >
< / object >
```

The XML element above is coded into three Prolog statements:

```
object(2,warrior).
position(2,1.4,2.3).
holdsitem(2,axe).
```

[1] The signals correspond to what in neuroscience is the concentration of certain chemical substances in human brain. The signal we use is a simplified representation of the concentration levels.

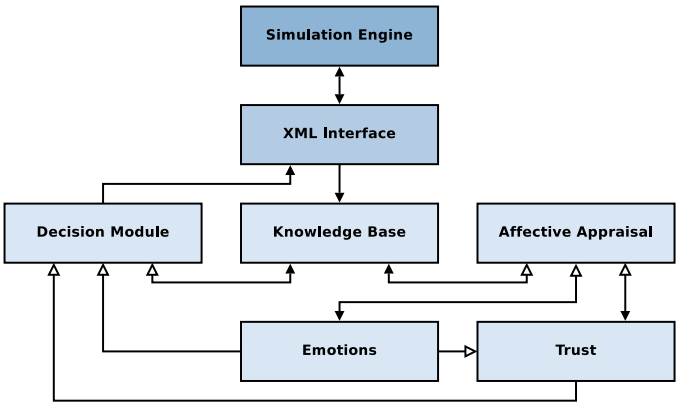


Figure 1: *Agent architecture*

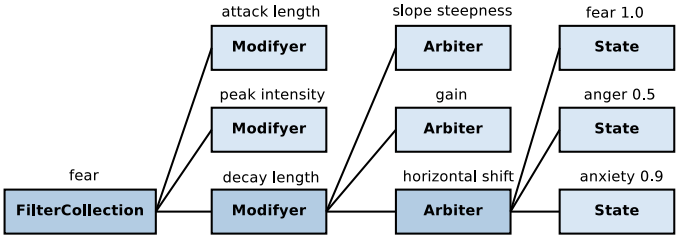


Figure 2: *Emotion filtering system*

The way the information is saved is highly similar to common database structures. Note that the first Prolog statement declares that there exists a warrior whose id is two. The remaining two statements, *position* and *holdsitem*, use the id of the warrior to save its position and the item held. This construction can be compared to the use of primary and foreign keys in databases. To retrieve all the objects currently holding an axe from the knowledge base the following Prolog query can be made:

```
?- object(X,_), holdsitem(X,axe).
```

The result of the query is then expressed in XML to be used by the other agent's modules:

```
< object id="2" type="warrior" >
      < holdsitem type="axe" / >
< / object >
```

The knowledge base saves the source of the information along with the information itself (this extra argument is not shown in the example above). The source in our system is an agent, either the agent itself (the event was perceived by the agent itself) or another agent. The source of information is important when determining the trust towards another agent. Should an agent provide false information it is likely to be trusted less in the future.

The time of the information acquisition is also stored in the knowledge base (not seen in example). This time of acquisition can also be used to simulate forgetfulness or the reliability of a piece of information.

3.4 Affective Appraisal Module

The main task of the Affective Appraisal module is to do an emotion appraisal, that is, an affective estimation of the incoming/perceived events. Depending on the events, different emotions will then be triggered. For instance, if an agent sees a wolf running towards it, the appraisal will trigger the emotion fear. The rules defining which emotion to trigger are easily customizable via XML rules:

```
< rule >
   < condition >
      < less_than threshold="80.0" >
         < new_value_from_memory variable_name="?MinDis" >
            < min_distance id1="#self" type="wolf"
            returnId="?WolfId" returnDis="?MinDis" / >
         < / new_value_from_memory >
      < / less_than >
   < / condition >
   < do >
      < trigger_emotion emotion="fear" >
      < / trigger_emotion >
   < / do >
< / rule >
```

This rule states that if the distance to the nearest wolf is less than 80 length units, then the emotion fear must be triggered.

3.4.1 Triggering Secondary Emotions

When the Affective Appraisal module encounters a rule that requests the triggering of the emotion disappointment, sadness should be triggered as well. It would also be appropriate to decrease the value of happiness. Triggering these kinds of secondary emotions is done automatically in the Affective Appraisal module. To decrease

happiness due to an increase in disappointment, one needs to create an emotional signal with a negative peak value. When adding this signal to the already existing happiness signals the overall value for happiness will decrease. The triggering of secondary emotions is configured through an XML document and the values can easily be tweaked for a better behavior.

3.4.2 Expectations

The appraisal module also handles the creation and management of expectations and (to some extent) trust. The handling of expectations is automatic and therefore hard-coded. An expectation in our agent architecture consists of several parts (defined in [Johansson 2007]):

Parameter	Explanation
expectation	The description of the event that is expected to happen.
time frame	The estimated time frame in which the event is predicted to happen
control-factor	The perceived control over the situation.
benevolence	The perceived goodness of the event seen from the agent's point of view.
probability	The estimated probability that the event will occur.

The expectations themselves are stored in the knowledge base. The affective appraisal module also checks for all the unfulfilled expectations and consequently the emotions that should be triggered in response.

3.5 Trust Module

The trust module[2] handles the agent's trust towards other agents (be they virtual or human). This module also specifies the trust-relevant personality traits of the agent. The agent can be trusting, suspicious, reactive, forgiving, etc. Personality traits can easily be changed over time to reflect the experience of the agent. For instance, if the agent has decided to trust other agents several times in the past but it always ends up badly, then the agent will likely not be so trusting in the future.

Our trust module builds upon several different trust models, in particular the ones in [Mayer et al. 1995; Zaheer et al. 1998]. It consists of four parts: *ability*, *reliability*, *predictability* and *integrity*. The overall trust is calculated as a sum of these four values. The values can also be used on their own if necessary. Different events will possibly trigger changes in these values. For instance, if an agent refuses to share information on the whereabouts of food, the other agents will most likely lower the integrity value of that agent.

One interesting aspect of this module is that it takes into consideration the values of emotions when calculating the perceived trust value for another agent. It has been shown that emotions influence our trust towards other people [Dunn and Schweitzer 2005]. This is especially true when we are not consciously aware of the source of our emotions. The way emotions affect trust is defined as:

[2]Note that trust is not used here to make decisions (this happens in the decision module). The level of trust is calculated and maintained here, and trust values are provided to other modules when needed.

$$
\begin{aligned}
overall\ trust &= T + EI \\
T &= ability + integrity + \\
&\quad reliability + predictability \\
EI &= \tfrac{OPC}{5} + \tfrac{WC}{10} + \tfrac{PC}{20} \\
OPC &= gratitude - anger \\
WC &= happiness - sadness \\
PC &= pride - shame
\end{aligned}
$$

where T is emotionless trust (the trust without any influence from emotions), EI is emotional influence, OPC is other-person control, WC is world control, PC is personal control. The definition of OPC, WC and PC is based on the work of [Dunn and Schweitzer 2005]. Since our system does not store information about the source of emotions yet, we do not have to be concerned about source salience which would otherwise affect the emotional influence on trust.

3.6 Decision Module

The task of the decision module is to select an action that the agent needs to perform at every time stamp. Choosing the best action is a complex task that depends on the history of previous events, knowledge and the current goals/objectives of the agent. At the moment, the decision module only consists of a simple set of XML rules similar to that of the Affective Appraisal module.

All emotions have to be triggered in the Appraisal module. However, expectations can be triggered in the decision module as they are often linked to a chosen action. In fact, when the agent chooses an action, it will most likely expect a certain outcome.

Actions that can be taken are physical actions (such as running, eating or sending messages to other agents), as well as internal actions (like triggering expectations). The physical actions available are determined by the simulation engine that will execute them.

4 Future Developments

Several improvements are needed to create a better agent architecture. We describe the planned extensions in detail below.

4.1 Decision Module

The current decision module uses only simple if-then rules. The first encountered rule with true body in the decision module determines the action to be executed next. Such a behavior is very limited, and we really need to extend this module. We are considering using a behavior network based approach [Maes 1989; Maes 1991] extended with emotions.

4.2 Action Management

A current diploma work [Zakaria (work in progress)] focuses on action management which deals with the actions once they have been selected. This includes script enabling and physics-based animation among other things. This module is placed slightly outside the agent architecture since it is not entirely a cognitive process but instead a more physics based process.

4.3 Learning

Another important capability that we want to ascribe to our agents is the ability to learn both from observation and experience. We are going to first develop/exploit a suitable learning algorithm and test it in our agent architecture.

4.4 Perception Module

The perception of the agent is influenced by its current emotional state, its preferences and its prior knowledge. An example of emotional influence is when an agent is very afraid of a predator and therefore pays little attention to other things around it. This module is very complex but it would also give great rewards in form of a more realistic human behavior system.

4.5 Memory Management

The agent cannot remember everything forever due to memory issues. It is also inhuman to remember information for too long. Humans remember some information for a very long time, while other things are forgotten after a short while (that is, human exploits short and long term memory). Some pieces of information are only remembered under certain conditions, such as when something in the vicinity reminds us of a remembered event (associative memory). Memories should also fade gradually over time, and become less and less certain. The aim of Memory Management module is to deal with all of these issues.

5 Virtual Reality Exposure in the Treatment of Psychiatric/Psychological Disorders

Over the last few years, therapists have started to use virtual reality (VR) exposure in the treatment of psychiatric and psychological disorders as a complement to standard therapy treatments, e.g. cognitive-behavioral therapy (CBT). VR seems to be an effective alternative to these traditional treatments and seems to bring significant advantages by allowing exposure to several different situations that would be difficult/costly to recreate in real life. In this section, we outline a number of research approaches that use VR exposure in the treatment of psychiatric disorders focussing on social disorders and phobias.

5.1 Social Disorders

Several studies have been conducted regarding the use of virtual reality in the treatment of social disorders, all leading to the conclusion that virtual reality seems adequate for such treatments (see [Grillon et al. 2006] p.105 for a list of supporting references). In fact, virtual reality can offer relevant scenarios which may not be easily available in real life. For example, it would be expensive to repeatedly take a patient on an airplane in order to treat him against the fear of flight.

A psychiatric disorder that can benefit from virtual reality based clinical treatment is social phobia. [Klinger et al. 2006] for example carried out research studies using VR in the treatment of fear of public speaking. Their findings show that human subjects are sensitive to virtual environments, and that the efficacy of VR treatment compares well with traditional CBT.

A number of research groups attempted to use VR-based treatments for social skills training for people with autistic spectrum disorders (ASDs). [Tartaro and Cassell 2006] proposed the use of authorable virtual peers (AVPs) to help children with ASD since they often lack communication and social interaction skills[3]. AVPs are animated characters capable of interacting and responding to children's input. In order to do so, AVPs incorporate three interaction modes - interact, control and author - that scaffold three key interaction practices - rehearse, observe and construct. The authors also embedded into their approach techniques of collaborative storytelling.

In addressing the use of virtual environments for social skills training for people with ASD, [Sarah et al. 2006] argue that there is still a lack of realism of virtual characters in scenes, although virtual environments have progressed quite a lot in terms of how realistic the scenes appear. It seemed clear to users that the virtual figures did not have any personality but they were instead preprogrammed. Therefore the authors advocated to make the characters more human-like.

One interesting interactive application is to employ characters in virtual learning environments (VLEs). Such an application can be used for example to assist children aged 8-11 years in dealing with social problems of bullying and mobbing in schools. In fact, VLEs populated with virtual characters offer children a safe environment where they can explore and learn via experiential activities. [Hall et al. 2004] carried out a similar project by using a software package called FearNot. In their project, FearNot presented the children with a bullying scenario with few virtual characters. Based on the children' choices taken during a simple dialogue with one of the characters, the narrative of the story could have had different endings. The objectives of this project was to teach children the strategy to use to solve a specific drama. Or if the children had selected strategies that frequently did not work, the educational message to them would have been to tell the drama to someone they trust.

5.2 Phobias

Phobias are common forms of anxiety disorders. Phobias are typically treated by using gradual exposure therapy that consists in exposing patients to anxiety-provoking stimuli in a gradual order with the aim to attenuate their anxiety. If the anxiety decreases, a more fearful situation is created. Traditionally, those stimuli are looked for in actual physical situations (in vivo) or by having the patient imagine the stimulus (in vitro). VR and virtual environments allow for an alternative/complementary option of exposure therapy. VR has been already used in treating specific phobias such as fear of heights, fear of spiders, fear of flying and claustrophobia, as well as agoraphobia (see for example [VR_Phobias 1999; VR_Med_Center 2007]).

[Takacs and Simon 2007] describe their clinical experience in deploying VR therapy for treating a variety of psychological disorders including depression, age-related conditions, pain distraction, anger-management as well as common phobias. They address three major aspects of rehabilitation:

cognitive rehabilitation - where visual stimuli and exercise in virtual environments are carried out to help patients regain their cognitive capabilities (e.g. after a stroke);

psychological rehabilitation - to help patients overcome fear, phobias and side-effects associated with stress;

physical rehabilitation - designed to make patients interact with virtual objects and carry out exercise in virtual space.

The authors state that they have successfully used the system, and present clinical validity results.

[3] It is reported that children with ASD show affinity for computers [Hart 2005]

To summarize, there is evidence from clinical data that VR can help to address a number of psychological disorders, and that there is an increasing number of health-care applications where VR can play a relevant role. Furthermore, VR-based therapy techniques offer a number of advantages over traditional ones, for example, they give the possibility of treating different kinds of phobias, stimuli in virtual environments can be manipulated, and patients can be confronted with their fears in environments that are felt safer than the real one.

6 Discussion

Once the characters' architecture is fully implemented, we aim at developing applications that build upon it. In particular, we aim at conceiving and developing interactive applications to be made publicly available. Such applications can be developed for different public sectors ranging from entertainment to education and health. Our primary objective is both to test the agent architecture and to develop concrete applications that can be of benefit in society. We expect that deploying complex forms of reasoning and learning as well as advanced forms of perception (for example, the ones that will incorporate techniques to acquire, interpret and select sensory information) will enhance realistic behavior modelling of virtual characters. Furthermore, ascribing emotions and personality traits to virtual characters will allow us to define complex forms of human-computer interaction.

One interesting test bed for our system would be to apply virtual characters and storytelling to create improvised dramas in a virtual school. Our vision is that Hall's original project (see Section 5.1) could be expanded in a number of ways, for example, to exploit better and more advanced types of human-computer interaction as well as to provide children with more realistic scenarios. This could be possible by exploiting our results and implemented characters' architecture. Of course, this must be done along side with cognitive psychologists, medical doctors, and school teachers.

Another challenging application where we can exploit VR-based technology is one that addresses the problem of facial expression recognition. It is well-documented that facial expression perception is impaired for example in individuals diagnosed with autism or Asperger's syndrome (see [Hobson et al. 1988; Phillips 2004]). In fact, autistic people do not appear to be able to pick up facial signals and notice other people's emotions. They can't read the signals or facial expressions of emotions in a normal way. Naturally, not being able to do so affects their social interactions. Recently, an interactive computer software program called FaceSay[4] has been shown to improve the ability of children with autism spectrum disorders (ASD) to recognize faces, facial expressions and emotions, according to the results of a study conducted by psychologists at the University of Alabama at Birmingham (UAB) [ScienceDaily Jun, 2007]. We believe that by ascribing virtual characters cognitive capabilities, perception and emotions we can go one step further. In fact, we can link the virtual character's facial expressions to its internal emotional state. Furthermore, since the emotional state depends upon the virtual character's cognitive and perceptive capabilities, we can relate the facial expression recognition problem to the problem of understanding the virtual character's cognitive/mental state. For example, we can associate the gesture of smiling (denoting happiness) to the fact of having successfully carried out a certain action. In this way, the understanding of the character's mental state will help in recognizing its emotional state, and vice versa. We hope that by establishing this association will help autistic people to enhance their overall ability in recognizing emotions and ultimately to improve their social interaction skills.

One major criticism to the use of VR-based technology for treatments is that the cost of this technology may be very high, and therefore its availability will remain limited (at least for the time being). Furthermore, some patients may experience simulation sickness. However, we already have VR technology and equipments available at our institution that we can freely use.

[4]Created by Symbionica L.L.C., features interactive games that let children with ASD practice recognizing the facial expressions of an avatar. Specifically, the computer game teaches the children where to look for facial cues such as an eye gaze or a facial expression. FaceSay is available at www.facesay.com

Acknowledgements

The authors would like to thank Monica Tavanti for her comments on an earlier version of the paper.

References

DUNN, J. R., AND SCHWEITZER, M. E., 2005. Feeling and Believing: The Influence of Emotion on Trust. Journal of Personality and Social Psychology, Vol. 88.

ESBJÖRNSSON, J., 2007. EMO - A Computational Emotional State Module. Master Thesis in Advanced Computer Graphics, Dept. of Science and technology (ITN), Linköping University. Available at http://www.ep.liu.se.

FUNGE, J. D., TU, X., AND TERZOPOULOS, D. 1999. Cognitive Modelling: knowledge, reasoning, and planning for intelligent characters. In *Proc. 26th annual conf. on Computer Graphics and Interactive Techniques (SIGGRAPH'99)*, ACM Press/Addison-Wesley Publishing Co., 29–38.

FUNGE, J. D. 1998. *Making Them Behave - Cognitive Models for Computer Animation*. PhD thesis, University of Toronto.

FUNGE, J. D. 1999. *AI for Games and Animation: A Cognitive Modelling Approach*. A. K. Peters, Wellesley, Mass.

GRILLON, H., RIQUIER, F., HERBELIN, B., AND THALMANN, D. 2006. Use of virtual reality as therapeutic tool for behavioural exposure in the ambit of social anxiety disorder treatment. Procs. 6th Int. Conf. Disability, Virtual Reality and Associated Technologies (ICDVRAT06), ISBN 07-049-98-653.

HALL, L., WOODS, S., AND DAUTENHAHN, K. 2004. Research Findings from Synthetic Character Research: Possible Implications for Interactive Communication with Robots. In *Proc. of the 2004 IEEE Int. W. on Robot and Human Interactive Communication*, IEEE, 53–58.

HART, M., 2005. Autism/Excel Study. Procs. 7th Int. ACM SIGACCESS Conf. on Computers and Accessibility, pp.136–141. ISBN 1-59593-159-7.

HOBSON, R., OUSTON, J., AND LEE, A. 1988. What's in a face? the case of autism. *Br J Psychol 79*, 441–453.

JOHANSSON, A., 2007. Modelling Expectations and Trust in Virtual Agents. Master Thesis in Media Technology, Dept. of Science and technology (ITN), Linköping University. Available at http://www.ep.liu.se.

KLINGER, E., LEGERON, P., ROY, S., CHEMIN, I. ANDM LAUER, F., AND NUGUES, P. 2006. Virtual reality exposure in the treatment of social phobia. Cybertherapy - Internet and Virtual Reality as Assessment and Rehabilitation Tools for Clinical Psychology and Neuroscience, G. Riva, C. Botella, P. Lgeron and G. Optale (Eds.) Amsterdam, IOS Press.

MAES, P. 1989. How to do the right thing. *Connection Science Journal, Special Issue on Hybrid Systems 1(3)*, 291–323.

MAES, P. 1991. A bottom-up mechanism for behavior selection in an artificial creature. In *Proceedings of the first International Conference on Simulation of Adaptive Behavior*, MIT Press, J. A. Meyer and S. Wilson, Eds.

MAYER, R. C., DAVIS, J. H., AND SCHOORMAN, F. D., 1995. An Integrative Model of Organizational Trust. Academy of Management Review, Vol. 20, No. 3, pp. 709-734.

PERLIN, K., AND GOLDBERG, A. 1996. Improv: A system for scripting interactive actors in virtual worlds. *Computer Graphics 29*, 3.

PHILLIPS, M. L. 2004. Editorial - facial processing deficits and social dysfunction: how are they related? *Brain - a Journal of Neurology 127*, 8, 1691–1692.

REITER, R. 1991. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness result for Goal Regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honour of John McCarthy*, Academic Press, V. Lifschitz, Ed., 359–380.

REYNOLDS, C. W. 1987. Flocks, herds, and school: A distributed behavioral model. In *Proc. Computer Graphics (SIGGRAPH'87)*, M. C. Stone, Ed., vol. 21, 25–34.

SARAH, P., LEONARD, A., AND MITCHELL, P. 2006. Virtual environments for social skills training: comments from two adolescents with autistic spectrum disorder. *Computer and Education 47*, 2, 186–206.

SCIENCEDAILY. Jun, 2007. Science News - Computer Game Helps Autistic Children Recognize Emotions. Available at www.sciencedaily.com.

TAKACS, B., AND SIMON, L. 2007. A clinical virtual reality rehabilitation system for phobia treatment. In *Proc. 11th Int. Conf. Information Visualization (IV'07)*, IEEE Computer Society, 798–806.

TARTARO, A., AND CASSELL, J. 2006. Authorable Virtual Peers for Autism Spectrum Disorders. W. on Language-Enabled Educational Technology and Development and Evaluation of Robust Spoken Dialogue Systems at the 17th European Conf. on Artificial Intelligence (ECAI06).

TERZOPOULOS, D. 1999. Artificial Life for Computer Graphics. *Communication of the ACM 42*, 8.

TU, X., AND TERZOPOULOS, D. 1994. Artificial Fishes: Physics, locomotion, perception, behavior. In *Proc. Computer Graphics (SIGGRAPH'94)*, ACM SIGGRAPH, ACM Press, A. Glassner, Ed., 43–50.

TU, X. 1999. *Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior*. PhD thesis, ACM Distinguished Ph.D Dissertation Series, LNCS 1635.

VR_MED_CENTER, 2007. The Virtual Reality Medical Center. Web site http://vrphobia.com.

VR_PHOBIAS, 1999. Virtual Reality and Phobias. Research project between Delft University of Technology and University of Amsterdam, The Netherlands. Web site http://graphics.tudelft.nl.

XSB-PROLOG. 2007. XSB Inc. Available at http://xsb.sourceforge.net/.

ZAHEER, A., MCEVILY, B., AND PERRONE, V., 1998. Does Trust Matter? Exploring the Effects of Inter organizational and Interpersonal trust on Performance. Organization Science, Vol 9, pp. 141–159.

ZAKARIA, R., (work in progress). Physics-Based Animation and Action Selection in Virtual Characters. Master Thesis (in progress) in Advanced Computer Graphics, Dept. of Science and technology (ITN), Linköping University. Expected Spring 2008.